DOCARENA: FROM RAW DOCUMENTS TO SEARCH TRAINING DATA BEYOND ANSWER CORRECTNESS

Anonymous authors
Paper under double-blind review

ABSTRACT

Search agents need to locate and combine information from document collections through multi-step retrieval and reasoning. We study how existing documents can provide both answer targets and evidence labels for learning this capability. However, correct question-answer pairs do not ensure accurate evidence labels, and incorrect labels can reward irrelevant retrieval or penalize the retrieval of supporting evidence. To address this challenge, we propose DocArena, a fully automated pipeline that constructs questions, answers, and automatically checked evidence-page labels from raw documents without human training annotations or expert search demonstrations. Specifically, we extract textual and visual page content with a multimodal LLM and profile the distribution of facts across pages to guide evidence selection and multi-page question construction. The pipeline checks answer consistency and tests whether each question remains answerable after removing each evidence page in turn. We construct DocArena-79K with $79,623$ QA pairs from $8,336$ documents across 16 domains and 49 languages. We further develop Doc-Search, which decouples multimodal page perception from a text-based policy and uses the constructed answer and retrieval targets for reinforcement learning. Experiments cover six multimodal document scenarios and seven text-based QA benchmarks. Pipeline ablations show that evidence checks improve training sample quality. Under fixed training settings, DocArena data improve document QA. Further analyses show that retrieval supervision improves evidence coverage and reduces repeated queries, while combining it with answer supervision improves QA. The trained policy also transfers from document collections to text-based QA.

1 INTRODUCTION

Knowledge-intensive applications need to answer questions over existing document collections, such as research reports, institutional archives, and business documents. The required information can be distributed across pages, tables, and figures, while a single retrieval may return only part of the evidence. Search agents (Jin et al., 2025b; Zheng et al., 2025; Chen et al., 2025d) address this need by formulating queries, reasoning over retrieved content, and using new clues to decide whether to search further or produce an answer. The agent needs to learn how to incorporate retrieved information into subsequent queries and combine evidence for answering. This need goes beyond document access and calls for search capability.

Existing methods build this capability by learning from successful search demonstrations (Zhang et al., 2026a; Chu et al., 2026) or practicing search through problem solving (Jin et al., 2025b; Song et al., 2025; Chen et al., 2025d). These methods involve search processes generated by a teacher model or explored by the policy model. Another practice directly supervises the retrieval process. For example, StepSearch (Wang et al., 2025b) improves multi-step search through feedback on evidence acquisition, with supervision constructed from existing multi-hop QA annotations. However, applying this practice to raw document collections requires training data to meet specific conditions. Following this line, we study how to automatically construct search training data from raw documents, with *questions, answers, and their supporting evidence generated and checked together*.

Constructing these training samples is non-trivial. First, multi-page questions should require facts from different pages and contain search clues without revealing answers or page identifiers. Simple

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

QA generation may overlook these cross-page dependencies. Second, evidence-page labels must identify the pages that jointly support an answer without including irrelevant or redundant pages. Correct QA pairs do not automatically constitute correct retrieval training samples. An answer can be valid even when its evidence labels are wrong, leading to rewards for irrelevant retrieval or penalties for retrieving supporting evidence. Third, evidence coverage and answer correctness need to be considered together. Evidence coverage does not ensure answer correctness. We therefore examine whether these labels improve evidence retrieval and reduce repeated queries during search.

To address these challenges, we propose **DocArena**, an automated pipeline that constructs questions, answers, and automatically checked evidence-page labels from raw documents for search agent training (Fig. 1). Specifically, we first use an MLLM to extract page content, including text and visual elements. We then profile the distribution of facts across candidate pages to guide evidence selection and question construction with reasoning templates. The pipeline checks answer consistency and tests whether a question remains answerable after removing each evidence page. These checks address answer correctness and evidence labeling separately before either label is used to train the agent. The pipeline requires no human-authored training questions, evidence labels, or search demonstrations. Each retained sample provides an answer target and a page-level retrieval target, while the policy generates its own search trajectory during training.

With the proposed pipeline, we construct DocArena-79K from CCpdf (Turski et al., 2023), containing $79,623$ QA pairs from $8,336$ documents across 16 domains and 49 languages. Among these samples, 92.3% involve multiple evidence pages, and 63.2% of evidence pages contain tables, figures, or charts. Evidence sets range from 1 to 20 pages, with a mean of 2.99 pages per sample. To assess the contribution of the data, we compare DocArena-79K with Search-R1 training data under the same policy initialization, retriever, rewards, and training schedule. Under this fixed training configuration, DocArena-79K improves document QA on MMLongBench-Doc and SlideVQA.

To train agents with these data, we develop Doc-Search, combining a text-based LLM policy, a multimodal page retriever, and a reading tool. This design decouples page perception from the policy and enables text-based LLMs to search multimodal documents. Training uses answer labels for final responses and evidence labels for retrieved pages. Experiments cover six document evaluation scenarios and seven text-based QA benchmarks. We find that evidence coverage and answer correctness are related but distinct training objectives. Retrieval supervision improves evidence coverage and reduces repeated queries, while its combination with answer supervision improves QA. Besides, the retrieval reward weight adjusts query repetition, and the training search budget affects how much QA accuracy improves with additional inference turns. Increasing training data further improves evidence retrieval, with larger gains on the evaluated groups requiring more pages. The resulting policy also transfers from document collections to a text-based knowledge environment. These findings support learning search behavior from document evidence without prescribing a reference search sequence. We summarize the contributions of this work as follows.

- This work studies how existing knowledge assets can provide training targets for search behavior. We propose DocArena, a fully automated pipeline that jointly constructs and checks questions, answers, and evidence-page labels without human training annotations or expert search demonstrations.
- We curate DocArena-79K, a training dataset spanning multiple domains and languages, with multi-page questions and evidence grounded in textual and visual document content.
- We develop Doc-Search to train text-based policies for multimodal document search and QA using the constructed answer and retrieval targets.
- Experiments evaluate answers and evidence labels, show higher Recall and EM with retrieval feedback, and QA gains under fixed training settings. Further analyses show reduced query repetition, differences in search budget use, and transfer to text-based QA.

2 Related Works

Document Understanding. Documents consist of heterogeneous elements such as tables, charts, figures, and complex layouts, which require models to jointly capture semantics and structure (Livathinos et al., 2025; Ouyang et al., 2025; Yang et al., 2025; Li et al., 2020; Wu et al., 2025d; Tian et al., 2025). Recent methods (Xiao et al., 2025; Wang et al., 2025c; Zhu et al., 2025; Chen

et al., 2025c; Tanaka et al., 2025; Caffagni et al., 2025) have advanced document understanding and retrieval, *e.g.*, DocLLM (Wang et al., 2024a) and DocLayLLM (Liao et al., 2025) improve the text–spatial alignment, while TextMonkey (Liu et al., 2024) uses an OCR-free vision-based approach. For long-document QA, SV-RAG (Chen et al., 2025b), Doc-React (Wu et al., 2025b), MDocAgent (Han et al., 2025), and M3DocRAG (Cho et al., 2024) explore MLLM-based self-retrieval, reasoning decomposition, or multi-agent collaboration for long-document QA. Reasoning-Table (Lei et al., 2025) applies reinforcement learning to the structured table reasoning. Doc-Dancer (Zhang et al., 2026a) synthesizes document-grounded tasks and successful search trajectories for supervised training. It learns from generated trajectories. By comparison, we construct evidence targets to evaluate the agent's own retrieval actions.

Agentic Search. We study search agents (Jin et al., 2025b;a; Gao et al., 2025; Zhang et al., 2025; Wu et al., 2025c;a; Zheng et al., 2025; Fan et al., 2025; Hu et al., 2026; Miroyan et al., 2026; Han et al., 2024) for multimodal, multi-page, reasoning-intensive document retrieval and QA. Training methods differ in how they supervise search behavior. Among them, Search-R1 (Jin et al., 2025b) is among the first to apply outcome-based rewards for incentivizing multi-turn search and reasoning, primarily focusing on text-based scenarios. ZeroSearch (Sun et al., 2025) replaces online search with a generator-based simulator to control context noise and improve training stability. IKEA (Huang et al., 2025) designs a knowledge-boundary aware reward to balance internal and external knowledge utilization during search. AutoRefine (Shi et al., 2025) introduces explicit knowledge refinement steps between retrieval rounds to distill and organize retrieved evidence, primarily evaluated on text-based QA benchmarks. O^2-Searcher (Mei et al., 2025) addresses open-domain open-ended QA by combining flexible retrieval and generation strategies. Besides, R-Search (Zhao et al., 2025) leverages multi-reward reinforcement learning, designed primarily for open-domain text-based scenarios. StepSearch (Wang et al., 2025b) derives step-wise search supervision from MuSiQue and rewards evidence acquisition. We take a step further by studying how raw document collections can provide reliable targets for retrieval supervision.

Training Environment for Search Agents. The concept of training environments for LLM RL has been established in the reasoning domain, where environments are defined by problem generators and verifiable rewards (Yu et al., 2025; Stojanovski et al., 2025). In our setting, $(q, a, \mathcal{E})$ triplets specify the question, answer target, and evidence targets. The document collection and retrieval tools define the interaction environment. Recent search agent post-training methods have benefited from improving the training environment, *e.g.*, IKEA (Huang et al., 2025) classifies samples into easy/hard splits via model probing, teaching agents when to search. R1-Searcher (Song et al., 2025) grades difficulty by rollout count and stages training accordingly. ZeroSearch (Sun et al., 2025) collects query-document pairs from real search engines to train a simulation model. REDSearcher (Chu et al., 2026) synthesizes multi-hop tasks from knowledge graph topologies with evidence dispersion control. Its post-training includes SFT on generated trajectories and RL with answer-correctness rewards. ASearcher (Gao et al., 2025) iteratively augments seed questions through injection and fuzzing mechanisms. SearchGym (Zhang et al., 2026b) jointly generates a knowledge graph, an aligned document corpus, and search tasks for simulated training. Overall, recent methods already automate task construction. By contrast, DocArena automatically constructs answer and retrieval targets from existing document collections and checks them with model-based tests. These targets provide feedback on evidence acquisition without requiring expert search demonstrations.

3 METHOD

3.1 PRELIMINARIES

Search Agent. A search agent interacts with an external search engine to answer a given question. At each turn, the agent formulates a search query, incorporates the retrieved results, and performs reasoning. Search agents (Shao et al., 2024; Jin et al., 2025b; Zheng et al., 2025) can be trained with outcome rewards without reference search trajectories. We construct $(q, a, \mathcal{E})$ tuples to provide answer targets and page-level retrieval targets. The policy generates its own search trajectories during training.

Notations. We instantiate the search agent in the multimodal document setting, a more complex and practically relevant scenario where the retrieval corpus consists of visually rich PDF documents. A document d consists of pages $\{p_1, \ldots, p_N\}$, and we adopt $\mathcal{P}$ for the full page collection across all

documents. Each page p has a rendered image p^{img} and extracted metadata p^{meta}. A training sample is a triplet $(q, a, \mathcal{E})$ consisting of a question q, a ground-truth answer a, and an evidence page set $\mathcal{E} \subset \mathcal{P}$. During training, the agent generates rollout trajectories given q.

3.2 DocArena: Automated Curation Pipeline

Training Targets for Search Behavior. The answer a provides an outcome target. The evidence pages $\mathcal{E}$ provide a target for information acquisition. Before using $\mathcal{E}$ for training, we check its support for the generated answer. We vary reasoning templates and source documents to construct a range of search tasks. Page images and extracted content retain access to both visual and textual information.

Pipeline Overview. As shown in Fig. 1, DocArena constructs questions, answer targets, and page-level retrieval targets from raw documents. These training labels are generated and checked automatically. Given a collection of raw documents $\mathcal{C} = \{d_1, d_2, \ldots, d_D\}$, the pipeline outputs training samples $\{(q_i, a_i, \mathcal{E}_i)\}$ through four stages (see Fig. 1): (1) document structuring and indexing, (2) cross-page information distribution profiling, (3) reasoning chain construction, and (4) cascaded quality assurance.

Stage I: Document Structuring and Indexing. We perform an offline, one-time preprocessing over the entire source document collection, producing three representations shared by all subsequent stages. (1) *Page images*—each page is rendered as an image, enabling downstream MLLM perception and interaction with multimodal tools. (2) *Structured semantic text*—an MLLM extracts body text, table content, figure descriptions, etc., from each page image into a per-page knowledge base. (3) *A dense retrieval index*—page-level semantic text is encoded by a text retriever into a FAISS inner-product index, serving the pipeline's internal cross-page retrieval needs.

Stage II: Cross-Page Information Distribution Profiling. We aim to select an evidence set that supports the answer and contains no page judged redundant by the verifier. We profile facts within a same-document candidate set to guide question construction and evidence selection. Stage IV then verifies answer consistency (Layer 2) and the necessity of every selected page (Layer 3).

Specifically, we sample a seed page p_s, retrieve related pages from the FAISS index, and retain same-document pages to form R (K=15$\sim$30). We select a subset $R_{\text{prof}} \subseteq R$ of at most 15 pages for profiling. An MLLM extracts *factual units* from each page in R_{prof}. Each factual unit is a minimal, self-contained knowledge point with a standardized `key` that categorizes the factual unit to the concept level. We aggregate units by `key` and compute each concept's *distribution width*

$$w(c) = |\mathcal{P}(c)|, \quad \text{where } \mathcal{P}(c) = \{p \in R_{\text{prof}} : c \text{ appears in } p\}. \tag{1}$$

The distribution width $w(c)$ partitions all concepts into three bands with distinct roles. *Page-specific concepts* ($w(c) = 1$) occur on exactly one page of R_{prof} and serve as retrieval anchors: every question must use at least one of them, so that the question corresponds to its evidence pages rather than to other candidate pages (*evidence exclusivity*). *Linking concepts* ($w(c) \in \{2, 3\}$) connect pages when present. *Broad concepts* ($w(c) > 3$) provide shared context that makes the verbalized question readable.

Stage III: Evidence-Grounded Reasoning Chain Construction. Given the distribution profile from Stage II, we use a sequence of MLLM calls to jointly select evidence pages, construct a reasoning chain, and verbalize the final QA pair. The evidence page selection is constrained by the distribution profile: at least one page-specific factual unit (w=1 within R_{prof}) must be included, and the selected pages must span at least M distinct pages within the same document. Evidence pages are selected from R, while $w(c)$ is computed on R_{prof}. Based on the selected pages and their factual units, the MLLM chooses a reasoning template from several predefined types, such as `chain`, `star`, `block_tree`, and `constraint_puzzle`, and constructs a reasoning skeleton. We maintain a global template counter and trigger re-selection when any single template exceeds a predefined threshold, preventing reasoning type collapse. Finally, the skeleton is verbalized into a natural language question and answer. The question states the primary anchor, i.e., the page-specific fact with the lowest $w(c)$, with its specific identifiers (Section A.2.7), but does not reveal page identifiers or contain the answer. Evidence pages for each generated question come from one source document. The agent searches the full document collection during training and evaluation.

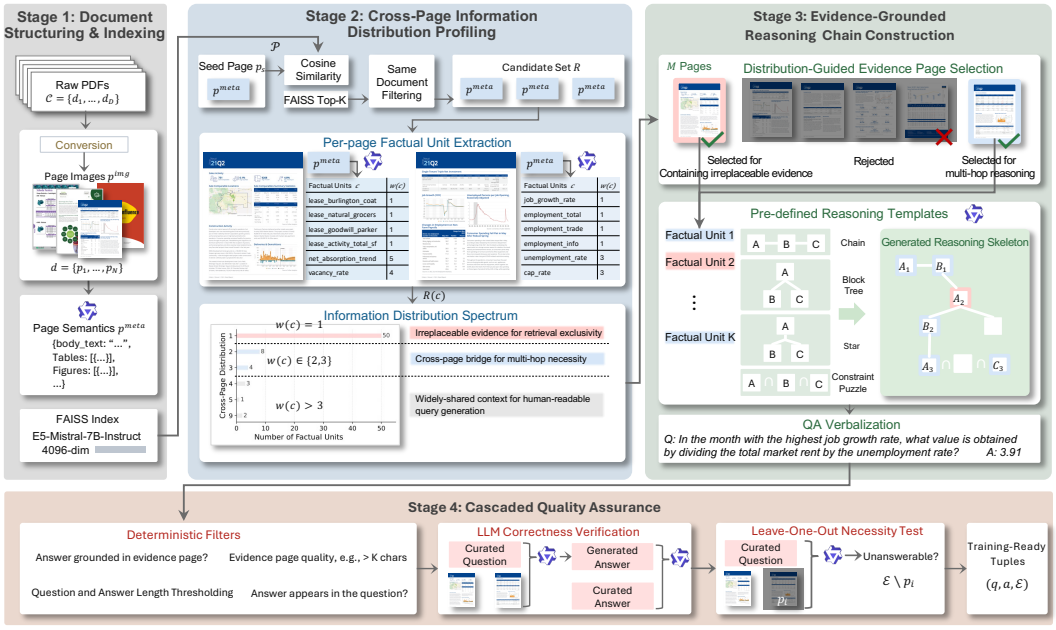


Figure 1: Overview of the proposed DocArena data curation pipeline. **Stage 1** converts raw PDFs into page images, MLLM-extracted structured text, and a dense retrieval index. **Stage 2** profiles facts across candidate pages to guide evidence selection. **Stage 3** constructs diverse, reasoning-intensive QA pairs grounded in the distribution profile with template-controlled diversity. **Stage 4** filters generated samples through three layers of automatic checks.

Stage IV: Cascaded Quality Assurance. DocArena also delivers insights on MLLM-driven data quality assessment, without any human annotation. We design a three-layer verification system whose checks are applied sequentially, and a candidate is discarded at the first check it fails.

Layer 1: Deterministic filters are rule-based checks including (a) does the answer contain facts not present in the evidence pages? (b) does the evidence page contain sufficient content (rejecting boilerplate pages with text below a character threshold)? (c) do the question and answer meet length constraints? (d) does the answer already appear in the question? *Layer 2: MLLM correctness verification.* An MLLM answers the question given the evidence pages, and a separate MLLM call judges whether its response is consistent with the curated answer. Samples where the MLLM cannot reproduce the answer are rejected. *Layer 3: Leave-one-page-out necessity test.* For each evidence page $p_j \in \mathcal{E}$, we remove it and test whether the question remains answerable from $\mathcal{E} \setminus \{p_j\}$ alone. A sample is retained only if the verifier cannot answer q from $\mathcal{E} \setminus \{p_j\}$ for every $p_j \in \mathcal{E}$, i.e., every labeled page is required to answer q. Both checks run within the same-document candidate set; pages elsewhere in the collection are not labeled, and retrieving them is not penalized by the retrieval reward.

Curated Training Dataset. We apply the proposed pipeline to CCpdf (Turski et al., 2023), a large-scale corpus of visually rich PDF documents, and verify by URL and file-hash matching that no source document appears in the evaluation benchmarks. The curated dataset, DocArena-79K, contains $79,623$ QA pairs from $8,336$ source documents, covering $37{,}345$ unique evidence pages with $238,271$ total page references. Among all samples, 92.3% require multi-page evidence (mean 2.99 pages per sample, range $1 \sim 20$). The evidence pages span diverse content, with 63.2% containing at least one table, figure, or chart beyond plain text. The source documents cover 16 content domains (engineering, medicine, education, finance, etc.) with no single domain exceeding 11%, and 10 document types (technical reports, scientific papers, manuals, forms, etc.) across 49 languages. Dataset distributions appear in Fig. 7, with further details in Sec. A.2.9.

3.3 The Doc-Search Agent Infrastructure

Infrastructure. As shown in Fig. 9, the infrastructure consists of a text-based LLM policy f_ϕ, a multimodal retriever ColPali (Faysse et al., 2025), and a reading tool. We use a text-based LLM

Table 1: Performance comparison on **MMLongBench-Doc** for both Single-page (SP) and Multi-page (MP) scenarios (Section A.4.1). Baselines are publicly released checkpoints; all methods use the same retriever, reading tool, prompt, and inference budget. Best results are **bolded** and second best are underlined.

Method	Single-page						Multi-page						
	Retrieval			QA Metrics			Retrieval				QA Metrics		
	Top-1 ↑	Top-5 ↑	NrDup ↓	EM ↑	Model ↑	PNLS ↑	Recall ↑	Precision ↑	F1 ↑	NrDup ↓	EM ↑	Model ↑	PNLS ↑
Search-R1 (Jin et al., 2025b)	69.82	79.67	14.68	25.94	23.63	34.81	55.24	32.18	37.09	22.77	20.10	17.92	28.98
DeepResearcher (Zheng et al., 2025)	69.61	81.93	38.84	25.50	20.81	33.23	55.82	33.37	38.25	44.39	19.71	16.25	27.16
IKEA (Huang et al., 2025)	69.82	81.52	19.85	28.30	**24.64**	37.26	54.43	**34.27**	38.41	23.62	21.03	18.89	30.02
R-Search (Zhao et al., 2025)	63.45	75.15	21.28	26.02	21.50	35.25	49.58	30.79	34.84	30.33	18.90	16.02	28.56
ZeroSearch (Sun et al., 2025)	62.63	76.18	31.03	24.37	21.05	33.44	48.49	31.33	34.74	26.32	18.73	15.89	28.14
AutoRefine (Shi et al., 2025)	65.71	76.80	14.81	26.60	22.29	35.84	52.23	32.66	36.75	15.04	20.47	17.50	29.82
StepSearch (Wang et al., 2025b)	63.86	77.00	3.82	25.13	21.17	35.09	52.09	30.31	35.24	5.30	20.96	17.30	30.50
ASearcher (Gao et al., 2025)	56.06	63.24	41.99	15.88	12.76	22.36	45.38	25.52	29.87	46.78	14.46	12.18	21.30
ReSearch (Chen et al., 2025d)	70.64	82.96	31.25	22.86	19.11	30.85	56.53	33.45	38.22	45.30	18.41	15.80	26.22
Ours	**75.77**	**85.22**	**2.46**	**29.34**	24.27	**38.27**	**61.38**	33.76	**39.68**	**2.97**	**24.18**	**19.43**	**33.12**

Table 2: Performance comparison on **VisRBench** FigureQA and TableQA (both single-page). Baselines are publicly released checkpoints; all methods use the same retriever, reading tool, prompt, and inference budget. Best results are **bolded** and second best are underlined.

Method	Figure QA						Table QA					
	Retrieval			QA Metrics			Retrieval			QA Metrics		
	Top-1 ↑	Top-5 ↑	NrDup ↓	EM ↑	Model ↑	PNLS ↑	Top-1 ↑	Top-5 ↑	NrDup ↓	EM ↑	Model ↑	PNLS ↑
Search-R1 (Jin et al., 2025b)	77.70	90.41	12.95	6.04	11.24	12.36	54.25	74.81	14.85	30.05	31.17	36.17
DeepResearcher (Zheng et al., 2025)	78.18	91.13	27.59	9.79	11.10	18.09	52.47	73.03	35.00	33.44	31.75	38.85
IKEA (Huang et al., 2025)	75.54	91.61	8.40	7.69	13.72	14.09	54.11	74.13	19.65	31.74	33.14	37.91
R-Search (Zhao et al., 2025)	73.38	84.65	18.12	11.02	14.42	16.24	47.40	68.29	28.71	31.24	30.66	37.00
ZeroSearch (Sun et al., 2025)	73.38	86.81	18.52	10.59	13.90	16.52	45.36	65.89	18.66	31.82	30.96	38.07
AutoRefine (Shi et al., 2025)	75.30	88.25	7.74	8.38	11.83	14.79	50.20	71.23	15.37	32.47	32.20	39.30
StepSearch (Wang et al., 2025b)	77.22	90.65	**2.35**	10.62	12.74	16.32	30.44	43.78	8.25	25.22	24.75	31.18
ASearcher (Gao et al., 2025)	58.03	64.75	29.79	11.00	13.06	**18.25**	46.64	61.08	52.46	26.21	25.43	32.80
ReSearch (Chen et al., 2025d)	79.62	96.40	23.56	8.38	11.07	16.15	53.98	75.95	25.22	32.19	31.72	38.34
Ours	**86.09**	**97.60**	2.84	**13.95**	**14.95**	17.50	**64.38**	**82.73**	**3.43**	**35.99**	**34.40**	**40.78**

for multi-step reasoning (Cao et al., 2025). At each step t, the agent generates a search query $\tilde{q}_t$, and ColPali retrieves the most relevant page images from the document collection $\mathcal{B}$. A reading tool returns text for each retrieved page. During training, the reading tool returns the page semantics p^{meta} from Stage I; at evaluation, the same checkpoint is used without retraining with the reader of each benchmark (Section A.3.3). The agent continues reasoning with the updated context. This process repeats until the agent produces a final answer or reaches the maximum search turn budget. During training, retrieved content tokens are masked from the policy loss following previous works (Jin et al., 2025b). Alternative reading tools improve QA on VisRBench FigureQA (Table 9).

Reward Design. We use a to score the final answer and $\mathcal{E}$ to score the pages retrieved at each search turn. The retrieval term provides feedback on information acquisition without prescribing a reference query sequence. We have the reward

$$R(\tau) = \lambda_{\text{QA}} \cdot r^{\text{QA}}(\tau) + \lambda_{\text{PR}} \cdot \sum_{t=1}^{T} \gamma^{t}\, r_t^{\text{PR}}, \tag{2}$$

where $r^{\text{QA}} = \mathbb{1}[\hat{a} = a]$ is the exact-match QA reward, r_t^{PR} is the recall of the pages newly retrieved at search turn t against $\mathcal{E}$ (Section A.3.1), and γ is a discount factor. The QA term rewards a correct answer regardless of which pages are retrieved. The retrieval reward is computed solely from page-level recall against $\mathcal{E}$ (Eq. 3); no query-level similarity term or duplicate penalty is used during training. NrDup (Section A.5.4) is an evaluation-only diagnostic and is never optimized. The progress reward is assigned to the search action token for fine-grained credit assignment. We provide more details in the supplementary.

4 Experiment

4.1 Experimental Settings

Evaluation Benchmarks. We evaluate the search agents on six multimodal document scenarios and seven text-based QA benchmarks. For multimodal document evaluation, we adopt MMLongBench-Doc (Ma et al., 2024), VisRBench (Chen et al., 2025a), and SlideVQA (Tanaka et al., 2023). VisRBench contains 53K QA pairs across $\sim 1,286$ documents; we evaluate on three sub-scenarios: FigureQA, TableQA, and TextQA, each requiring different types of visual-structural reasoning. MMLongBench-Doc contains $1,082$ expert-annotated questions with both single-page (SP) and multi-page (MP) scenarios. We evaluate $2,215$ SlideVQA questions over slide decks. For text-based QA, we adopt seven benchmarks following previous works (Jin et al., 2025b), *e.g.*, Natural

Table 3: Performance comparison on **VisRBench** TextQA (single-page) and **SlideVQA** (multi-page). Baselines are publicly released checkpoints; all methods use the same retriever, reading tool, prompt, and inference budget.

Method	VisRBench TextQA (SP)						SlideVQA (MP)					
	Retrieval			QA Metrics			Retrieval				QA Metrics	
	Top-1 ↑	Top-5 ↑	NrDup ↓	EM ↑	Model ↑	PNLS ↑	Recall ↑	Precision ↑	F1 ↑	NrDup ↓	EM ↑	Model ↑	PNLS ↑
Search-R1 (Jin et al., 2025b)	74.34	91.44	9.99	43.83	44.40	47.68	81.81	53.85	61.04	16.01	43.55	40.12	49.62
DeepResearcher (Zheng et al., 2025)	70.68	87.09	27.56	48.21	44.73	50.15	78.17	52.11	58.91	36.63	41.32	36.79	45.75
IKEA (Huang et al., 2025)	73.73	89.83	10.64	44.27	45.84	48.24	76.44	52.62	58.68	9.55	43.01	39.84	48.48
R-Search (Zhao et al., 2025)	68.77	86.72	20.94	46.91	43.56	48.55	74.54	51.20	57.04	16.73	44.26	39.44	48.49
ZeroSearch (Sun et al., 2025)	56.01	71.42	34.44	43.70	40.98	48.05	64.00	45.17	49.67	8.97	40.81	36.67	47.14
AutoRefine (Shi et al., 2025)	70.40	87.26	11.09	47.58	45.82	51.39	75.16	50.23	56.68	9.59	42.84	38.43	47.73
StepSearch (Wang et al., 2025b)	68.34	87.88	**0.61**	**51.16**	47.65	**53.83**	80.82	52.49	60.01	**2.79**	47.90	**42.64**	52.11
ASearcher (Gao et al., 2025)	66.02	75.92	32.06	36.00	34.54	40.92	63.82	41.63	47.29	27.66	35.41	31.18	40.31
ReSearch (Chen et al., 2025d)	75.55	91.95	22.04	47.61	45.45	50.42	84.98	54.71	62.72	29.77	43.78	38.97	47.91
Ours	**83.27**	**94.98**	2.06	49.78	**47.93**	49.80	**86.28**	**59.78**	**66.95**	21.21	**47.96**	42.43	**52.18**

Table 4: Performance comparison on **text-based QA benchmarks**. Baselines are publicly released checkpoints; all methods use the same retriever and identical inference parameters. Our policy is post-trained on DocArena-79K without using the seven text QA benchmarks for training.

Method	Natural Questions ↑	TriviaQA ↑	PopQA ↑	HotpotQA ↑	2WikiMultiHopQA ↑	MuSiQue ↑	Bamboogle ↑	Avg ↑
Search-R1 (Jin et al., 2025b)	**39.20**	59.22	38.91	36.16	27.58	14.94	36.29	36.04
StepSearch (Wang et al., 2025b)	36.84	57.51	36.41	36.31	32.05	**19.53**	35.48	36.30
ZeroSearch (Sun et al., 2025)	36.29	57.09	35.79	32.11	34.90	10.88	29.03	33.73
R-Search (Zhao et al., 2025)	34.71	58.85	36.17	32.59	29.76	12.83	35.48	34.34
ReSearch (Chen et al., 2025d)	36.40	58.71	38.87	34.88	27.97	17.09	**40.32**	36.32
DeepResearcher (Zheng et al., 2025)	35.60	58.44	37.06	34.85	31.39	14.40	39.52	35.89
Ours	36.68	**59.63**	**40.34**	**37.23**	**36.82**	14.11	33.87	**36.95**

Questions (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), PopQA (Mallen et al., 2023), HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020), MuSiQue (Trivedi et al., 2022), and Bamboogle (Press et al., 2023), covering both single-hop and multi-hop reasoning.

Metrics. For single-page retrieval, we report Top-k accuracy (Top-1 and Top-5), measuring whether the evidence page appears in the retriever's top-k pages in any search turn. For multi-page retrieval, we compute recall, precision, and F_1 over the set of evidence pages. For QA quality, we adopt Exact Match (EM) (Yu et al., 2024), Partial Normalized Levenshtein Similarity (PNLS) (Chen et al., 2024), and a model-based score (Wang et al., 2025a). We additionally report the Near-Duplicate Rate (NrDup↓), which measures the percentage of multi-round queries where at least one pair has token-level Jaccard similarity > 0.8. NrDup measures lexical repetition among queries for samples with at least two search turns. We report it with evidence Recall and F1 to assess search behavior.

Scope of the comparison. Tables 1–4 compare released checkpoints under a shared inference interface (retriever, reader, prompt, budget). This is a system-level comparison: baselines were trained in text-based Wikipedia environments, and their differences in training data, rewards, and environment are retained. We therefore do not attribute these gaps to any single component. The contribution of the constructed data is isolated by controlled experiments that fix initialization, retriever, rewards, and schedule and vary only the training data (Table 6b): DocArena-79K vs. Search-R1 training data, and pipeline variants at equal scale (79,623 samples each). The contribution of retrieval supervision is isolated by the reward ablation (Table 5a).

Compared Methods. We compare with nine RL-trained search agents: Search-R1 (Jin et al., 2025b), DeepResearcher (Zheng et al., 2025), IKEA (Huang et al., 2025), R-Search (Zhao et al., 2025), ZeroSearch (Sun et al., 2025), AutoRefine (Shi et al., 2025), StepSearch (Wang et al., 2025b), ASearcher (Gao et al., 2025), and ReSearch (Chen et al., 2025d). We evaluate released policy checkpoints with the same ColPali retriever, reading tool, and inference settings. The checkpoints retain their original training differences. The data-source comparison separately fixes the model, retriever, rewards, and training configuration (Table 11). For all methods, we adopt publicly released checkpoints trained on Qwen2.5-7B-Instruct[1] (Team et al., 2024).

Implementation. We adopt Qwen2.5-7B-Instruct as the base policy model. We train with Group Relative Policy Optimization (GRPO) using a batch size of 512, group size of 5, and learning rate 1×10^{-6}. The maximum number of search turns during training is 2. The retriever returns top-3 relevant pages per query. The reward weights are set to $\lambda_{\text{QA}} = 1.0$ and $\lambda_{\text{PR}} = 0.5$ (Eq. 2). During inference, the maximum search turns is 2 for SP scenarios and 3 for MP scenarios. For text-based QA evaluation, we follow Search-R1 (Jin et al., 2025b) and use E5-base-v2 with the 2018 Wikipedia

[1]Qwen2.5-7B is adopted when the instructional checkpoint is not available.

Table 5: Ablation studies on (a) reward components and (b) PR reward ratio (MMLongBench-Doc MP). PR = progress-based retrieval reward, QA = outcome-based answer quality reward.

(a) Reward modeling

Reward	Retrieval				QA		
	Rec	Prec	F1	NrDup↓	EM	Mod	PNLS
w/o PR	53.68	**34.19**	38.49	36.36	20.37	17.23	28.29
w/o QA	**62.95**	30.74	37.19	5.57	5.88	4.49	12.04
PR+QA	61.38	33.76	**39.68**	**2.97**	**24.18**	**19.43**	**33.12**

(b) PR reward ratio

PR Ratio	Retrieval				QA		
	Rec	Prec	F1	NrDup↓	EM	Mod	PNLS
0.1	**61.93**	**33.87**	**39.81**	11.07	21.50	18.03	30.67
0.3	60.70	33.00	39.15	7.24	21.85	18.27	31.05
0.5	61.38	33.76	39.68	**2.97**	**24.18**	**19.43**	**33.12**

Table 6: Quality and training value of constructed targets. (a) Quality pass rates (%) on 100 generated samples per pipeline variant. (b) Doc-Search trained with the same policy initialization, retriever, rewards, and training schedule, varying only the training data. Pipeline variants use 79,623 samples each, generated from the same source documents. Full metrics for the Search-R1 data and DocArena-79K rows appear in Table 11.

(a) Pipeline ablations

Pipeline	GPT-5.4	Claude-Opus-4.7	Human
Full pipeline	99	100	100
w/o correctness	85	80	89
w/o exclusivity	79	76	82
w/o leave-one-page-out	33	29	39

(b) Training data comparison

Training data	MMLongBench-Doc MP			SlideVQA	
	Recall	F1	EM	F1	EM
Search-R1 data (∼169K)	57.55	39.53	20.08	64.19	44.79
Naive generation (79.6K)	56.69	37.63	19.25	64.32	43.05
w/o leave-one-page-out (79.6K)	59.72	37.81	21.39	65.59	45.41
DocArena-79K (full pipeline)	**61.38**	**39.68**	**24.18**	**66.95**	**47.96**

dump as the retriever for all methods. Our implementation is built on veRL (Sheng et al., 2024). Training is performed on 8×A100 GPUs.

4.2 Main Results

Multimodal Search and QA Performance. Tables 1∼3 compare search agents across six multimodal document scenarios. Under the shared inference interface, Doc-Search achieves the highest Top-1 accuracy or Recall in all six document settings, *e.g.*, on MMLongBench-Doc MP, Doc-Search obtains Recall 61.38 and EM 24.18, outperforming ReSearch by +4.85 on Recall and IKEA by +3.15 on EM. It has the highest EM in four settings and EM comparable to StepSearch on SlideVQA and TextQA. Our NrDup ranges from 2.06% to 3.43% on MMLongBench-Doc and VisRBench, and is 21.21% on SlideVQA, where the Base-initialized policy reaches 0.95% with comparable Recall (Table 13), indicating that lexical repetition on slide decks varies with the initial model.

4.3 Model Analysis

Reliability of Constructed Targets. Two annotators audited 100 randomly sampled examples and reported joint question–answer–evidence pass rates of 99% and 100% (Table 7). For 100 generated samples per pipeline variant, the human pass rate was 100% for the full pipeline and 89%, 82%, and 39% without correctness, exclusivity, and leave-one-page-out checks, respectively (Table 6a), using the joint criterion of Table 7. *w/o exclusivity* removes the distribution-guided evidence selection, so that QA pairs are constructed from the sampled pages without requiring a page-specific ($w(c)$=1) fact and need not be exclusive to their evidence pages; *w/o correctness* removes Layer 2, and *w/o leave-one-page-out* removes Layer 3.

Effects of Retrieval Supervision. Table 5a compares reward components on MMLongBench-Doc MP. Adding retrieval reward raises Recall from 53.68 to 61.38 and EM from 20.37 to 24.18, while reducing NrDup from 36.36% to 2.97%. Since r_t^{PR} counts only newly retrieved evidence pages, re-retrieving pages already found receives no reward, consistent with the reduction in NrDup. Retrieval reward alone gives Recall of 62.95 but EM of 5.88. Combining both rewards achieves the best F1 and QA metrics in this ablation. The constructed evidence targets thus provide useful training feedback beyond answer correctness.

Weight of Retrieval Supervision. Table 5b varies the PR reward ratio λ_{PR}. With the same training data, increasing λ_{PR} from 0.1 to 0.5 reduces NrDup from 11.07% to 2.97% and raises EM from 21.50 to 24.18. The constructed targets allow us to vary the emphasis on retrieval during training.

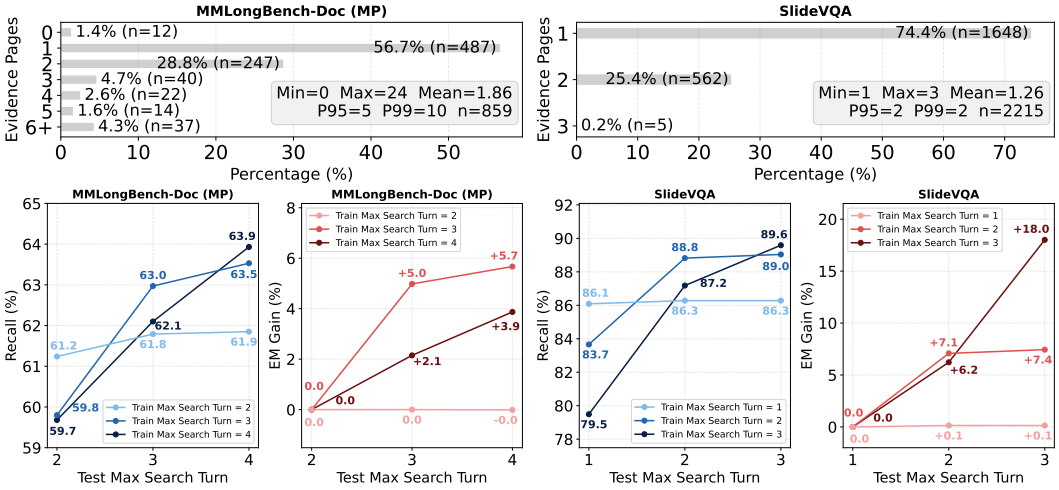


Figure 2: Effects of training and inference search budgets. The panels report retrieval Recall and EM gains on MMLongBench-Doc and SlideVQA. Each EM curve is measured relative to that model's smallest inference budget. Training budgets affect how agents use additional search opportunities.

Training Value of Constructed Targets. We compare Search-R1 data (Jin et al., 2025b) and DocArena-79K with the same policy initialization, retriever, reward configuration, and training schedule. DocArena-79K improves EM by 4.10 points on MMLongBench-Doc and 3.17 points on SlideVQA (Table 6b). This comparison supports the training value of the constructed targets. Full settings and metrics are reported in Section A.5.1.

Training Value of Evidence Checks. Table 6b also trains Doc-Search on 79,623 samples from each pipeline variant under the same configuration. Removing the leave-one-page-out check lowers all five metrics, e.g., EM from 24.18 to 21.39 on MMLongBench-Doc MP and from 47.96 to 45.41 on SlideVQA, and naive generation from the same documents lowers every metric further, e.g., EM to 19.25 and 43.05. The checks therefore improve the trained agent, not only sample quality (Table 6a).

Use of Additional Search Turns. Fig. 2 compares training and inference search budgets. Training budgets are $T \in \{1, 2, 3\}$ for the SlideVQA evaluation and $T \in \{2, 3, 4\}$ for the MMLongBench-Doc evaluation. Each EM curve reports gains relative to that model's smallest inference budget. On SlideVQA, two additional inference turns yield EM gains of $+0.1$, $+7.4$, and $+18.0$ for the $T = 1$, $T = 2$, and $T = 3$ policies. On MMLongBench-Doc, the $T = 4$ policy reaches Recall of 63.9 at four inference turns, but its EM gain is smaller than that of the $T = 3$ policy. Training search budgets affect how effectively agents use additional search opportunities.

Transfer to Text-Based Search. Table 4 compares the proposed methods with text-based search agents on seven text-based QA benchmarks. Doc-Search reaches an average EM of 36.95, comparable to the text-trained search agents. On the multi-hop benchmark 2WikiMultiHopQA, our method reaches 36.82, outperforming ZeroSearch (34.90) by $+1.92$ and Search-R1 (27.58) by $+9.24$. Further analyses examine training data scale (Sec. A.5.2) and policy initialization (Sec. A.5.3).

5 Conclusion

This work showed that existing document collections could provide supervision for learning how to search, beyond serving as sources for answering questions. We proposed DocArena, which made this supervision explicit by automatically constructing and checking answer and retrieval targets. The resulting supervision improved evidence acquisition and answering, and the trained policy transferred to a different knowledge environment. These findings supported a path for organizations to turn existing knowledge assets into search capability without manually constructing training labels or expert search demonstrations.

AI USE STATEMENT

In this work, we used generative AI tools for generating synthetic data sets, designing or providing feedback on research methodology or experiments, implementing methods, assisting with translation, cleaning and reformatting dataset, supporting qualitative and thematic data analysis, and interpreting results. Helping develop theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, proposing or refining hypotheses are not applicable to this work. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2. 5-vl technical report. In *arXiv preprint arXiv:2502.13923*, 2025.

Davide Caffagni, Sara Sarto, Marcella Cornia, Lorenzo Baraldi, and Rita Cucchiara. Recurrence-enhanced vision-and-language transformers for robust multimodal document retrieval. In *CVPR*, 2025.

Pengfei Cao, Tianyi Men, Wencan Liu, Jingwen Zhang, Xuzhao Li, Xixun Lin, Dianbo Sui, Yanan Cao, Kang Liu, and Jun Zhao. Large language models for planning: A comprehensive and systematic survey. *arXiv preprint arXiv:2505.19683*, 2025.

Jian Chen, Ruiyi Zhang, Yufan Zhou, Ryan Rossi, Jiuxiang Gu, and Changyou Chen. Mmr: Evaluating reading ability of large multimodal models. In *CoRR*, 2024.

Jian Chen, Ming Li, Jihyung Kil, Chenguang Wang, Tong Yu, Ryan Rossi, Tianyi Zhou, Changyou Chen, and Ruiyi Zhang. Visr-bench: An empirical study on visual retrieval-augmented generation for multilingual long document understanding. In *arXiv preprint arXiv:2508.07493*, 2025a.

Jian Chen, Ruiyi Zhang, Yufan Zhou, Tong Yu, Franck Dernoncourt, Jiuxiang Gu, Ryan A Rossi, Changyou Chen, and Tong Sun. Sv-rag: Lora-contextualizing adaptation of mllms for long document understanding. In *ICLR*, 2025b.

Jun Chen, Dannong Xu, Junjie Fei, Chun-Mei Feng, and Mohamed Elhoseiny. Document haystacks: Vision-language reasoning over piles of 1000+ documents. In *CVPR*, 2025c.

Mingyang Chen, Linzhuang Sun, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Haofen Wang, Jeff Z Pan, Wen Zhang, Huajun Chen, et al. Learning to reason with search for llms via reinforcement learning. *NeurIPS*, 2025d.

Jaemin Cho, Debanjan Mahata, Ozan Irsoy, Yujie He, and Mohit Bansal. M3docrag: Multi-modal retrieval is what you need for multi-page multi-document understanding. *arXiv preprint arXiv:2411.04952*, 2024.

Zheng Chu, Xiao Wang, Jack Hong, Huiming Fan, Yuqi Huang, Yue Yang, Guohai Xu, Chenxiao Zhao, Cheng Xiang, Shengchao Hu, et al. Redsearcher: A scalable and cost-efficient framework for long-horizon search agents. *arXiv preprint arXiv:2602.14234*, 2026.

Cheng Cui, Ting Sun, Suyin Liang, Tingquan Gao, Zelun Zhang, Jiaxuan Liu, Xueqing Wang, Changda Zhou, Hongen Liu, Manhui Lin, et al. Paddleocr-vl-1.5: Towards a multi-task 0.9 b vlm for robust in-the-wild document parsing. *arXiv preprint arXiv:2601.21957*, 2026.

Wei Fan, Wenlin Yao, Zheng Li, Feng Yao, Xin Liu, Liang Qiu, Qingyu Yin, Yangqiu Song, and Bing Yin. Deepplanner: Scaling planning capability for deep research agents via advantage shaping. *arXiv preprint arXiv:2510.12979*, 2025.

Manuel Faysse, Hugues Sibille, Tony Wu, Bilel Omrani, Gautier Viaud, Céline Hudelot, and Pierre Colombo. Colpali: Efficient document retrieval with vision language models. In *ICLR*, 2025.

Jiaxuan Gao, Wei Fu, Minyang Xie, Shusheng Xu, Chuyi He, Zhiyu Mei, Banghua Zhu, and Yi Wu. Beyond ten turns: Unlocking long-horizon agentic search with large-scale asynchronous rl. In *arXiv preprint arXiv:2508.07976*, 2025.

Rujun Han, Yuhao Zhang, Peng Qi, Yumo Xu, Jenyuan Wang, Lan Liu, William Yang Wang, Bonan Min, and Vittorio Castelli. Rag-qa arena: Evaluating domain robustness for long-form retrieval augmented question answering. In *EMNLP*, 2024.

Siwei Han, Peng Xia, Ruiyi Zhang, Tong Sun, Yun Li, Hongtu Zhu, and Huaxiu Yao. Mdocagent: A multi-modal multi-agent framework for document understanding. *CoRR*, 2025.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *COLING*, 2020.

Tiansheng Hu, Yilun Zhao, Canyu Zhang, Arman Cohan, and Chen Zhao. Sage: Benchmarking and improving retrieval for deep research agents, 2026.

Ziyang Huang, Xiaowei Yuan, Yiming Ju, Jun Zhao, and Kang Liu. Reinforced internal-external knowledge synergistic reasoning for efficient adaptive search agent. In *arXiv preprint arXiv:2505.07596*, 2025.

Bowen Jin, Jinsung Yoon, Priyanka Kargupta, Sercan O Arik, and Jiawei Han. An empirical study on reinforcement learning for reasoning-search interleaved llm agents. In *arXiv preprint arXiv:2505.15117*, 2025a.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. In *COLM*, 2025b.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *ACL*, 2017.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. In *TACL*, pp. 453–466, 2019.

Fangyu Lei, Jinxiang Meng, Yiming Huang, Tinghong Chen, Yun Zhang, Shizhu He, Jun Zhao, and Kang Liu. Reasoning-table: Exploring reinforcement learning for table reasoning. *arXiv preprint arXiv:2506.01710*, 2025.

Minghao Li, Yiheng Xu, Lei Cui, Shaohan Huang, Furu Wei, Zhoujun Li, and Ming Zhou. Docbank: A benchmark dataset for document layout analysis. In *COLING*, 2020.

Wenhui Liao, Jiapeng Wang, Hongliang Li, Chengyu Wang, Jun Huang, and Lianwen Jin. Doclayllm: An efficient multi-modal extension of large language models for text-rich document understanding. In *CVPR*, 2025.

Yuliang Liu, Biao Yang, Qiang Liu, Zhang Li, Zhiyin Ma, Shuo Zhang, and Xiang Bai. Textmonkey: An ocr-free large multimodal model for understanding document. In *CoRR*, 2024.

Nikolaos Livathinos, Christoph Auer, Maksym Lysak, Ahmed Nassar, Michele Dolfi, Panos Vagenas, Cesar Berrospi Ramis, Matteo Omenetti, Kasper Dinkla, Yusik Kim, et al. Docling: An efficient open-source toolkit for ai-driven document conversion. In *AAAI*, 2025.

Yubo Ma, Yuhang Zang, Liangyu Chen, Meiqi Chen, Yizhu Jiao, Xinze Li, Xinyuan Lu, Ziyu Liu, Yan Ma, Xiaoyi Dong, et al. Mmlongbench-doc: Benchmarking long-context document understanding with visualizations. In *NeurIPS*, 2024.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *ACL*, 2023.

Jianbiao Mei, Tao Hu, Daocheng Fu, Licheng Wen, Xuemeng Yang, Rong Wu, Pinlong Cai, Xinyu Cai, Xing Gao, Yu Yang, et al. O2-searcher: A searching-based agent model for open-domain open-ended question answering. In *arXiv preprint arXiv:2505.16582*, 2025.

Mihran Miroyan, Tsung-Han Wu, Logan King, Tianle Li, Jiayi Pan, Xinyan Hu, Wei-Lin Chiang, Anastasios N. Angelopoulos, Trevor Darrell, Narges Norouzi, and Joseph E. Gonzalez. Search arena: Analyzing search-augmented llms. In *ICLR*, 2026.

Linke Ouyang, Yuan Qu, Hongbin Zhou, Jiawei Zhu, Rui Zhang, Qunshu Lin, Bin Wang, Zhiyuan Zhao, Man Jiang, Xiaomeng Zhao, et al. Omnidocbench: Benchmarking diverse pdf document parsing with comprehensive annotations. In *CVPR*, 2025.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In *EMNLP*, pp. 5687–5711, 2023.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. In *arXiv preprint arXiv:2402.03300*, 2024.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*, 2024.

Yaorui Shi, Sihang Li, Chang Wu, Zhiyuan Liu, Junfeng Fang, Hengxing Cai, An Zhang, and Xiang Wang. Search and refine during think: Autonomous retrieval-augmented reasoning of llms. In *NeurIPS*, 2025.

Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. R1-searcher: Incentivizing the search capability in llms via reinforcement learning. In *arXiv preprint arXiv:2503.05592*, 2025.

Zafir Stojanovski, Oliver Stanley, Joe Sharratt, Richard Jones, Abdulhakeem Adefioye, Jean Kaddour, and Andreas Köpf. Reasoning gym: Reasoning environments for reinforcement learning with verifiable rewards. *NeurIPS*, 2025.

Hao Sun, Zile Qiao, Jiayan Guo, Xuanbo Fan, Yingyan Hou, Yong Jiang, Pengjun Xie, Yan Zhang, Fei Huang, and Jingren Zhou. Zerosearch: Incentivize the search capability of llms without searching. In *arXiv preprint arXiv:2505.04588*, 2025.

Ryota Tanaka, Kyosuke Nishida, Kosuke Nishida, Taku Hasegawa, Itsumi Saito, and Kuniko Saito. Slidevqa: A dataset for document visual question answering on multiple images. In *AAAI*, 2023.

Ryota Tanaka, Taichi Iki, Taku Hasegawa, Kyosuke Nishida, Kuniko Saito, and Jun Suzuki. Vdocrag: Retrieval-augmented generation over visually-rich documents. In *CVPR*, 2025.

Qwen Team et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2(3), 2024.

Yang Tian, Zheng Lu, Mingqi Gao, Zheng Liu, and Bo Zhao. Mmcr: Benchmarking cross-source reasoning in scientific papers. In *ICCV*, 2025.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Musique: Multihop questions via single-hop question composition. In *TACL*, volume 10, pp. 539–554, 2022.

Michał Turski, Tomasz Stanisławek, Karol Kaczmarek, Paweł Dyda, and Filip Graliński. Ccpdf: Building a high quality corpus for visually rich documents from web crawl data. In *ICDAR*, 2023.

Dongsheng Wang, Natraj Raman, Mathieu Sibue, Zhiqiang Ma, Petr Babkin, Simerjot Kaur, Yulong Pei, Armineh Nourbakhsh, and Xiaomo Liu. DocLLM: A layout-aware generative language model for multimodal document understanding. In *ACL*, 2024a.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. Improving text embeddings with large language models. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 11897–11916, 2024b.

Qiuchen Wang, Ruixue Ding, Yu Zeng, Zehui Chen, Lin Chen, Shihang Wang, Pengjun Xie, Fei Huang, and Feng Zhao. Vrag-rl: Empower vision-perception-based rag for visually rich information understanding via iterative reasoning with reinforcement learning. In *NeurIPS*, 2025a.

Ziliang Wang, Xuhui Zheng, Kang An, Cijun Ouyang, Jialu Cai, Yuhang Wang, and Yichao Wu. Stepsearch: Igniting llms search ability via step-wise proximal policy optimization. In *EMNLP*, 2025b.

Zining Wang, Tongkun Guan, Pei Fu, Chen Duan, Qianyi Jiang, Zhentao Guo, Shan Guo, Junfeng Luo, Wei Shen, and Xiaokang Yang. Marten: Visual question answering with mask generation for multi-modal document understanding. In *CVPR*, 2025c.

Haoran Wei, Yaofeng Sun, and Yukun Li. Deepseek-ocr 2: Visual causal flow. *arXiv preprint arXiv:2601.20552*, 2026.

Jinming Wu, Zihao Deng, Wei Li, Yiding Liu, Bo You, Bo Li, Zejun Ma, and Ziwei Liu. Mmsearch-r1: Incentivizing lmms to search. In *arXiv preprint arXiv:2506.20670*, 2025a.

Junda Wu, Yu Xia, Tong Yu, Xiang Chen, Sai Sree Harsha, Akash V Maharaj, Ruiyi Zhang, Victor Bursztyn, Sungchul Kim, Ryan A. Rossi, Julian McAuley, Yunyao Li, and Ritwik Sinha. Docreact: Multi-page heterogeneous document question-answering. In *ACL*, 2025b.

Weiqi Wu, Xin Guan, Shen Huang, Yong Jiang, Pengjun Xie, Fei Huang, Jiuxin Cao, Hai Zhao, and Jingren Zhou. Masksearch: A universal pre-training framework to enhance agentic search capability. In *arXiv preprint arXiv:2505.20285*, 2025c.

Xixi Wu, Yanchao Tan, Nan Hou, Ruiyang Zhang, and Hong Cheng. Molorag: Bootstrapping document understanding via multi-modal logic-aware retrieval. In *EMNLP*, 2025d.

Han Xiao, Yina Xie, Guanxin Tan, Yinghao Chen, Rui Hu, Ke Wang, Aojun Zhou, Hao Li, Hao Shao, Xudong Lu, et al. Adaptive markup language generation for contextually-grounded visual document understanding. In *CVPR*, 2025.

Zhibo Yang, Jun Tang, Zhaohai Li, Pengfei Wang, Jianqiang Wan, Humen Zhong, Xuejing Liu, Mingkun Yang, Peng Wang, Shuai Bai, et al. Cc-ocr: A comprehensive and challenging ocr benchmark for evaluating large multimodal models in literacy. In *ICCV*, 2025.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *EMNLP*, 2018.

Peng Yu, Zeyuan Zhao, Shao Zhang, Luoyi Fu, Xinbing Wang, and Ying Wen. Learning to reason in structured in-context environments with reinforcement learning. *arXiv preprint arXiv:2509.23330*, 2025.

Yue Yu, Wei Ping, Zihan Liu, Boxin Wang, Jiaxuan You, Chao Zhang, Mohammad Shoeybi, and Bryan Catanzaro. Rankrag: Unifying context ranking with retrieval-augmented generation in llms. In *NeurIPS*, 2024.

Haozhen Zhang, Tao Feng, and Jiaxuan You. Router-r1: Teaching llms multi-round routing and aggregation via reinforcement learning. In *NeurIPS*, 2025.

Qintong Zhang, Xinjie Lv, Jialong Wu, Baixuan Li, Zhengwei Tao, Guochen Yan, Huanyao Zhang, Bin Wang, Jiahao Xu, Haitao Mi, et al. Docdancer: Towards agentic document-grounded information seeking. *arXiv preprint arXiv:2601.05163*, 2026a.

Xichen Zhang, Ziyi He, Yinghao Zhu, Sitong Wu, Shaozuo Yu, Meng Chu, Wenhu Zhang, Haoru Tan, and Jiaya Jia. SearchGym: Bootstrapping real-world search agents via cost-effective and high-fidelity environment simulation. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 18633–18665, San Diego, California, United States, July 2026b. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.848. URL https://aclanthology.org/2026.acl-long.848/.

Qingfei Zhao, Ruobing Wang, Dingling Xu, Daren Zha, and Limin Liu. R-search: Empowering llm reasoning with search via multi-reward reinforcement learning. In *arXiv preprint arXiv:2506.04185*, 2025.

Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. Deepresearcher: Scaling deep research via reinforcement learning in real-world environments. In *EMNLP*, 2025.

Zhaoqing Zhu, Chuwei Luo, Zirui Shao, Feiyu Gao, Hangdi Xing, Qi Zheng, and Ji Zhang. A simple yet effective layout token in large language models for document understanding. In *CVPR*, 2025.

A Appendix

A.1 Overview

The supplementary material is organized as follows:

Section A.2: Data Curation Pipeline

- A.2.1 Page-Specific Anchors and Evidence Checks
- A.2.2 Pipeline Yield and Rejection Statistics
- A.2.3 Computational Cost Analysis
- A.2.4 MLLM Prompt Templates
- A.2.5 Stage I: Document Structuring and Indexing
- A.2.6 Stage II: Cross-Page Distribution Profiling
- A.2.7 Stage III: Reasoning Chain Construction
- A.2.8 Stage IV: Cascaded Quality Assurance
- A.2.9 Source Collection and Dataset Statistics

Section A.3: Doc-Search Agent Infrastructure and Training

- A.3.1 More Illustrations on Reward Design and Credit Assignment
- A.3.2 Training Hyperparameters
- A.3.3 Reading Tool Implementations
- A.3.4 More Illustrations on the Inference Prompt Template

Section A.4: Evaluation Benchmarks and Implementation Details

- A.4.1 More Illustrations on MMLongBench-Doc
- A.4.2 More Illustrations on VisRBench
- A.4.3 More Illustrations on SlideVQA
- A.4.4 More Illustrations on the Comparison
- A.4.5 Intra/Inter-Document Illustrations

Section A.5: Agent Behavior Analysis and Ablation Studies

- A.5.1 Training Data Source Ablation
- A.5.2 Data Scaling Curves
- A.5.3 Base Model Comparison
- A.5.4 NrDup Metric

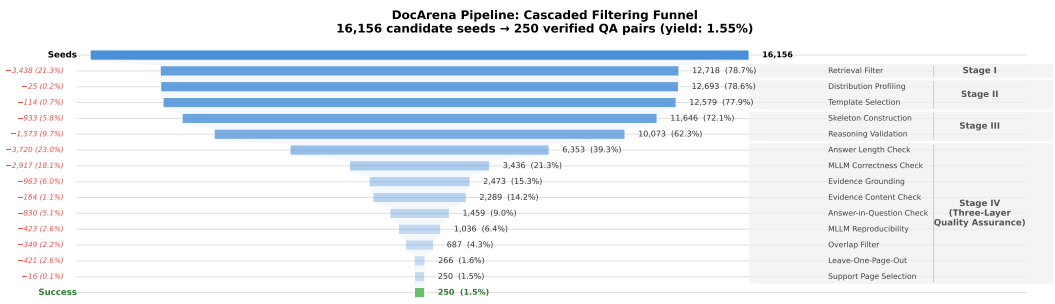


Figure 3: Cascaded filtering funnel of the DocArena curation pipeline. From 16,156 candidate seeds, each gate progressively filters low-quality samples, yielding 250 valid QA pairs (1.55% yield rate). Red annotations indicate the number and percentage of samples rejected at each gate.

A.2 Data Curation Pipeline

A.2.1 Page-Specific Anchors and Evidence Checks

Stage III selects evidence pages under the distribution profile from Stage II and requires the reasoning chain to use at least one $w(c)=1$ fact. Stage IV then tests the generated QA pair: the MLLM correctness verification (Layer 2) checks whether the answer can be reproduced from the evidence pages, and the leave-one-page-out test (Layer 3) checks whether every evidence page is required.

A.2.2 Pipeline Yield and Rejection Statistics

To characterize the pipeline's filtering behavior, we run the full curation pipeline on a separate analysis subset and track per-stage outcomes. We generate 250 verified QA pairs from in total of 16,156 candidate seed pages for analysis (yield rate 1.55%, 95% CI: $[1.36\%, 1.74\%]$). Fig. 3 visualizes the cascaded filtering funnel. The y-axis lists the 14 gates grouped by pipeline stage (Stages I–IV), and the bar length indicates the number of samples surviving after each gate. Red annotations show the count and percentage of samples rejected at each gate. The last row reports rejections at support page selection, the evidence page selection step (Section 3.2). The three largest rejection sources are the answer length filter (23.0%), the retrieval pre-filter (21.3%, documents with fewer than 3 same-document pages in the retrieval neighborhood), and the MLLM correctness verification (18.1%). The cascade filters candidates at several stages. The component ablation in Table 6a further shows how the checks affect final sample quality.

Fig. 4 (left) shows the reasoning template distribution among the verified QA pairs. The x-axis lists the four templates and the y-axis shows the proportion. `chain` (35.2%) and `block_tree` (36.4%) together account for over 70% of samples, while `constraint_puzzle` (14.4%) and `star` (14.0%) each contribute a smaller share. All four reasoning templates remain represented, and no single template accounts for more than 40% of the verified samples. Fig. 4 (right) shows the distribution width $w(c)$ of all 22,354 factual units extracted from the verified samples: 98.2% are page-specific ($w(c)=1$), 1.5% are linking ($w(c) \in \{2,3\}$), and 0.2% are broad ($w(c) \geq 4$). Page-specific facts are therefore abundant: only 25 of 16,156 seeds are rejected at distribution profiling (Fig. 3), so Stage III can select a primary anchor with $w(c)=1$ for nearly every candidate set. Cross-page dependence is enforced by the reasoning skeleton, in which each operator takes inputs from different pages and the final answer requires all selected pages (Section A.2.7); Layer 3 then verifies this dependence for every labeled page. The manual audit and pipeline ablations assess the quality of the resulting training targets (Tables 7 and 6a).

We use Qwen2.5-VL-32B-Instruct (Bai et al., 2025) throughout the curation pipeline. DocArena-79K and the reported pipeline statistics use this backbone. The pipeline can also use other MLLMs through the same extraction, generation, and verification interfaces. Filtering outcomes may vary with the backbone.

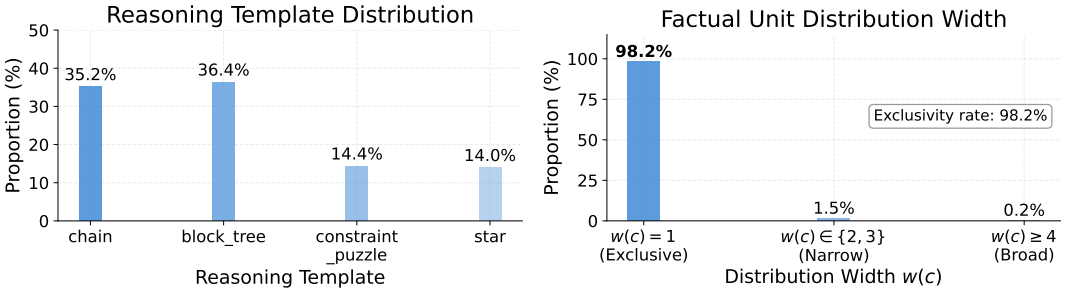


Figure 4: **Left**: Reasoning template distribution among valid QA pairs. **Right**: Distribution width $w(c)$ of factual units. Within the profiled candidate sets, 98.2% of factual units occur on one page (w=1).

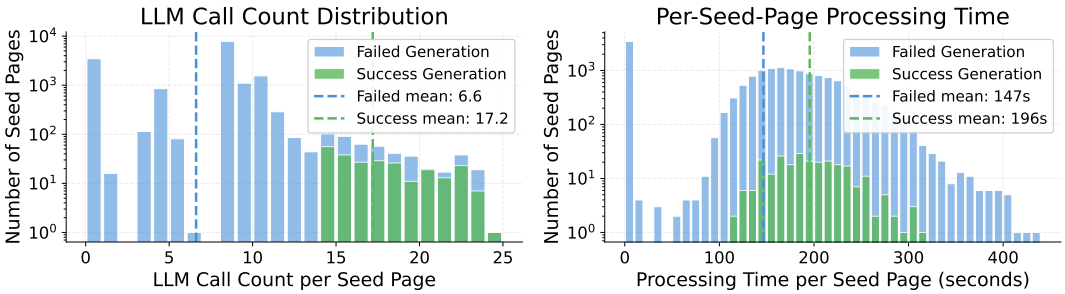


Figure 5: **Left**: MLLM call count per seed page for successful (green) vs. failed (blue) generations. Successful seed pages require more calls (17.2 mean) as they pass through all cascaded gates. **Right**: Per-seed-page processing time. The spike near 0s reflects seed pages rejected by the retrieval pre-filter without any MLLM call. Dashed lines indicate means.

A.2.3 Computational Cost Analysis

Fig. 5 presents the computational cost analysis of the curation pipeline. The left panel shows the distribution of MLLM call counts per seed page (x-axis) against the number of seed pages (y-axis, log scale), split into successful generation (green) and failed generation (blue). Successful seed pages require an average of 17.2 MLLM calls, while failed seed pages average 6.6 calls. Successful seeds pass all verification gates. Failed seeds stop at the first failed check, so some avoid later MLLM calls. The right panel shows the per-seed-page processing time with a similar split. Successful seed pages average 196s compared to 147s for failed ones. These means describe successful and failed seeds separately. Note that they are not the amortized cost per retained QA pair. A visible spike near 0s corresponds to seed pages rejected by the retrieval pre-filter without any MLLM call. notably, for the failed distribution, its tail extends beyond that of successful cases as some failed seed pages pass several gates of Stage IV before being rejected, accumulating processing time comparable to successful cases before eventually being rejected. The three-layer cascade (Section A.2.8) stops processing a candidate once a check fails. This avoids later verification calls for rejected candidates.

Fig. 6 breaks down the computational cost by pipeline stage and reasoning template. The left panel reports the mean processing time of each stage for successful (green) and failed (blue) generations. Retrieval (Stage I) takes less than 0.1s per seed page because it relies on a pre-built FAISS index lookup without any MLLM call. The four MLLM-intensive stages account for most of the cost: Distribution Profiling (∼70s), Template Selection (∼48s), Atom Extraction (∼32s), and Support Selection (∼16s). Across these stages, successful and failed cases have similar mean processing times. Validation is different: successful cases (22.1s) take 2.4× longer than failed cases (9.3s). This is expected, as a successful sample must pass all nine gates of Stage IV (grouped into the three layers of Section 3.2), while a failed sample exits at the first gate it does not pass, and the remaining gates are skipped.

The right panel shows the proportion of each reasoning template within each outcome group: blue bars represent the share of each template among all failed generations, and green bars represent the share among all successful generations. The red line (right axis) shows the yield rate per template. Both the failed and successful distributions are balanced across templates. Among seeds that reach

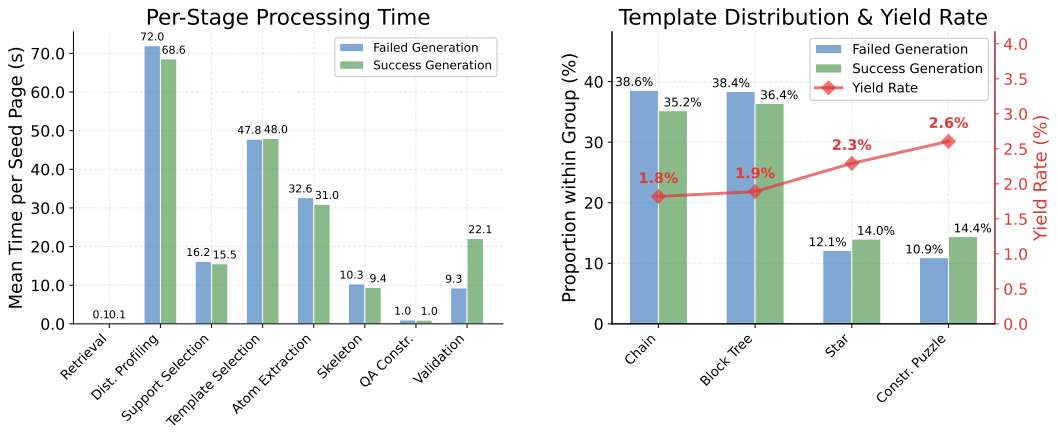


Figure 6: **Left**: Per-seed-page processing time by pipeline stage for successful (green) and failed (blue) generations. Retrieval takes < 0.1s. **Right**: Proportion of each reasoning template within each outcome group (blue: failed; green: successful), with yield rate per template (red line, right axis). Both success/fail distributions and yield rates (1.8–2.6%) are balanced across templates.

template selection, the yield rate is consistent, ranging from 1.8% to 2.6% across the four templates. All four templates remain represented after filtering, preserving a range of reasoning structures in the curated data.

A.2.4 MLLM Prompt Templates

The curation pipeline (Section 3.2 of the main paper) uses MLLM calls at multiple stages. All calls use Qwen2.5-VL-32B-Instruct (Bai et al., 2025). We list four representative prompt templates below, organized by pipeline stage. Variables enclosed in braces (e.g., {`text`}) are replaced with the actual content at runtime.

Stage II: Factual Unit Extraction.

For each page in the profiled subset R_{prof}, the following prompt extracts factual units for cross-page matching.

Factual Unit Extraction Prompt

Extract atomic factual units from this document page.

PAGE: {page_id}
CONTENT: {text}

For each factual unit, provide:
1. A concise description (one sentence)
2. The type: one of [entity_name, date, numeric_value, specification, procedure, requirement, condition, definition, classification, reference, relationship, other]
3. A short semantic grouping key for matching equivalent facts across pages (e.g. "product_temperature_range", "certification_standard"). The key must be general enough to match the SAME concept across different pages.
4. The specific text/value from this page (brief quote)

CRITICAL:
- Focus on SUBSTANTIVE content, not headers/footers/boilerplate
- Extract specific values, named entities, technical specifications, requirements
- Grouping keys must be concept-level, not document-specific (e.g. "operating_temperature_range" not "model_x100_temp_range")

Respond in JSON: {"factual_units": [{"description": ..., "type": ..., "grouping_key": ..., "value": ...}]}

Stage III: Reasoning Skeleton Construction.

This prompt builds the reasoning program using the selected template, operators, and anchor constraints.

Reasoning Skeleton Construction Prompt

Build a program using the '{selected_template}' template that combines atoms from {num_pages} pages.

Template: {selected_template}
Pages: {support_page_ids}
Extracted atoms: {atoms_summary}

ANCHOR POINTS (must be used in the reasoning chain): {anchor_summary}

RELEVANT INFORMATION DISTRIBUTION: {relevant_dist_json}

CRITICAL CONSTRAINTS:
1. Each operation must use inputs from DIFFERENT pages
2. The final answer must require information from ALL pages
3. No single page should contain all necessary information

NUMERICAL CALCULATION REQUIREMENTS:
- When building arithmetic operations, explicitly state the exact numbers from the pages
- Show step-by-step calculation with actual values (e.g., "25.5 + 30.2 = 55.7")
- Use only the specific numerical values found in the provided pages
- Do not estimate or approximate – use exact figures from the source material

Build a program that:
- Uses appropriate operators (JOIN, FILTER, AGG, LOOKUP, ARITH)
- Ensures cross-page dependencies
- Follows the {selected_template} structural pattern
- Produces a DERIVED answer that can only be determined by combining information from ALL pages
- The expected_answer must be concise: a specific value, name, identifier, short phrase
- The answer should be the result of cross-page information synthesis, not a direct quote from any single page

Respond with: {"program_steps": [...], "expected_answer": "...", "cross_page_verification": "..."}

Stage III: Question Verbalization.

This prompt converts the reasoning program into a natural language question with anchor stratification (Section A.2.7).

Question Verbalization Prompt

Convert this program into a natural language question that would lead to the answer: "{answer}"

Program steps: {program_steps}

PRIMARY ANCHOR (mention with specific identifiers): {primary_desc}

SECONDARY ANCHORS (reference implicitly): {secondary_descs}

REQUIREMENTS:
1. Do NOT mention specific page numbers or locations
2. The question should naturally require the information processing described in the program
3. Make it sound like a realistic question someone would ask
4. Ensure the question implies the need for multiple pieces of information

Generate a clear, natural question.

Stage IV Layer 3: Leave-One-Page-Out Test.

This prompt tests whether the question remains answerable when each evidence page is excluded. Pages are given as their Stage I semantics p^{meta}, the same representation the policy reads during training.

Leave-One-Page-Out Test Prompt

Based ONLY on the following pages, answer this question.
If the pages contain enough information to answer fully, provide a brief answer.
If the pages do NOT contain enough information, respond with exactly: UNANSWERABLE

Pages: {remaining_pages_text}

Question: {question}

Answer (or UNANSWERABLE):

A.2.5 Stage I: Document Structuring and Indexing

Each page is rendered at 144 DPI, matching the page-level input granularity of the ColPali retriever used during RL training.

The dense retrieval index in Stage I uses E5-Mistral-7B-Instruct (Wang et al., 2024b) as the text encoder. This index serves the pipeline's internal cross-page retrieval needs in Stage II and is separate from the ColPali retriever used by the Doc-Search agent at training and inference time. During data curation, the pipeline operates on structured semantic text extracted by the MLLM, so a text encoder can capture fine-grained semantic similarity between pages. During agent interaction, the retriever operates on raw page images and the agent's generated queries, so ColPali's vision-language alignment is more suitable. This separation also avoids coupling the pipeline's retrieval quality with the agent's retrieval quality: the pipeline index can be built once, while the agent retriever can be changed independently without re-running the curation pipeline.

A.2.6 Stage II: Cross-Page Distribution Profiling

Given a seed page, the pipeline retrieves from the global FAISS index and retains pages from the same document. The retained top-K pages form R. Distribution analysis uses a subset $R_{\text{prof}} \subseteq R$ of at most 15 pages. The width $w(c)$ is computed only within R_{prof}.

Documents are skipped if fewer than 3 same-document pages are retrieved or if fewer than 2 pages yield factual units. After aggregation, documents with no exclusive (w=1) or narrow ($w \in \{2, 3\}$) concepts are also skipped, as the distribution profile does not contain sufficient structure to support the intended cross-page question construction.

A.2.7 Stage III: Reasoning Chain Construction

The reasoning skeleton is constructed using five operators:

- `LOOKUP`: extract a value from a specific page.
- `JOIN`: associate information across pages.
- `FILTER`: apply a condition to select relevant entries.
- `AGG`: aggregate values (sum, average, count).
- `ARITH`: perform arithmetic ($+$, $-$, $\times$, $\div$, $\%$).

Each operator takes inputs from different pages, and the skeleton enforces that the final answer requires information from all selected evidence pages. For templates involving numerical calculations, the skeleton includes step-by-step computation with exact values from the pages.

During QA verbalization, the pipeline stratifies anchor points into a primary anchor (mentioned with specific identifiers in the question) and secondary anchors (referenced implicitly). The primary anchor is selected as the one with the lowest distribution width and the highest discriminative score, which is computed based on the presence of named entities, clause or section numbers, and numerical values in the anchor description. This stratification places a specific search clue in the question without explicitly including the answer.

The minimum number of evidence pages per sample M is configurable between 1 and 4. DocArena-79K uses M=2 for 92.3% of samples and M=1 for 7.7%, which provides single-page lookup

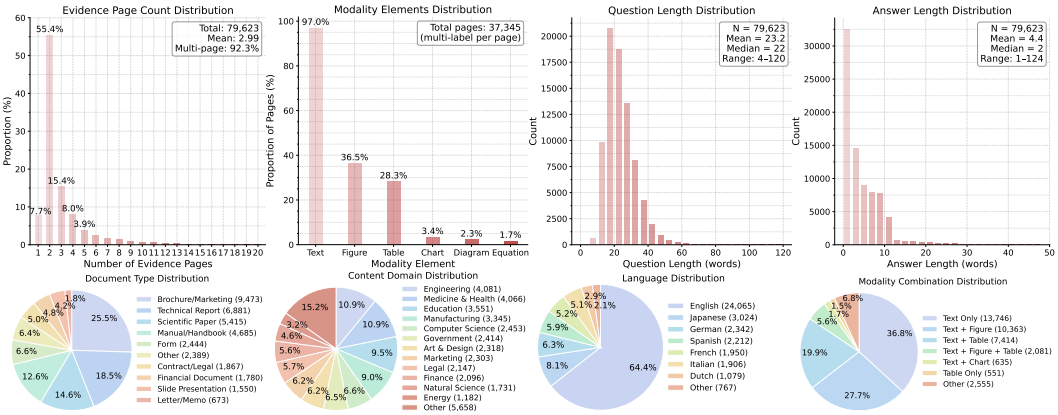


Figure 7: Dataset statistics of DOCARENA. **Top row**: distributions of evidence pages per question, modality elements per page, question length, and answer length. **Bottom row**: distributions of document type, content domain, language, and modality combination.

Table 7: Manual alignment audit on 100 randomly sampled examples. Values are pass rates (%). q–$\mathcal{E}$: the question is answerable from the evidence pages; $\mathcal{E}$–a: the evidence pages support the answer; q–a: the answer is correct for the question; Nec.: every evidence page is required to answer the question. Joint denotes the intersection of the four checks.

Annotator	q–$\mathcal{E}$	$\mathcal{E}$–a	q–a	Nec.	Joint
1	99	99	100	99	99
2	100	100	100	100	100

tasks; for $M{=}1$, the cross-page constraints of the skeleton prompt are disabled. Answer length is constrained to at most 15 words or 128 characters.

A.2.8 STAGE IV: CASCADED QUALITY ASSURANCE

More Discussions on MLLM Error Propagation. The curation pipeline uses MLLM calls in Stages II and III to extract factual units and construct reasoning chains. Errors can be introduced during these stages, and may be propagated into the generated QA pairs. The checks in Stage IV filter detected inconsistencies. Layer 1 applies deterministic filters that check whether the answer is grounded in the evidence text; QA pairs where the answer contains facts not present in the evidence pages are rejected. Layer 2 re-answers the question in a separate MLLM call and rejects inconsistent answers. Layer 3 rejects a sample when the verifier can still answer after a page removal. The effect of each layer is assessed, independently of the pipeline backbone, by two external judges (GPT-5.4 and Claude-Opus-4.7) and human annotators (Table 6a).

A.2.9 SOURCE COLLECTION AND DATASET STATISTICS

Manual Audit of Training Targets. Two annotators independently assessed 100 randomly sampled DocArena-79K examples with access to the full source documents, agreeing on 99 of 100 samples. Table 7 reports alignment among the question (q), answer (a), and evidence pages ($\mathcal{E}$). The joint pass rates were 99% and 100%, averaging 99.5%. The audit supports the alignment of the automatically constructed training targets.

CCpdf (Turski et al., 2023) is a large-scale, multilingual corpus of visually rich PDF documents sourced from Common Crawl. The corpus contains 1.1 million documents with 14.5 million pages, with an average of 12.9 pages per document. The documents are sourced from diverse websites across the Internet, covering business, legal, scientific, and technical domains, with the majority created after 2010. Compared to prior publicly available document corpora such as IIT-CDIP (6.5M documents from a single domain in the 1990s) and OCR-IDL (4.6M single-domain documents), CCpdf provides broader domain and language coverage with longer documents on average. The

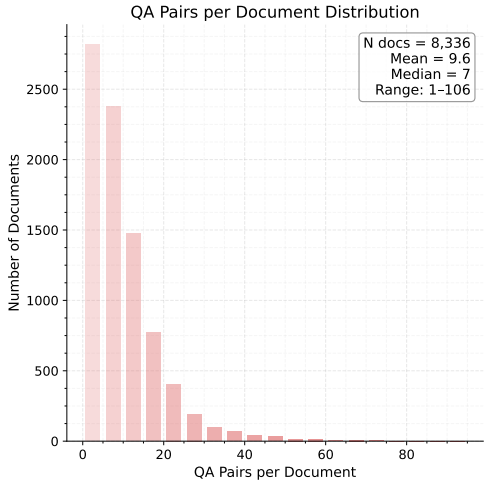


Figure 8: Distribution of QA pairs per document in DocArena-79K (mean 9.6, median 7).

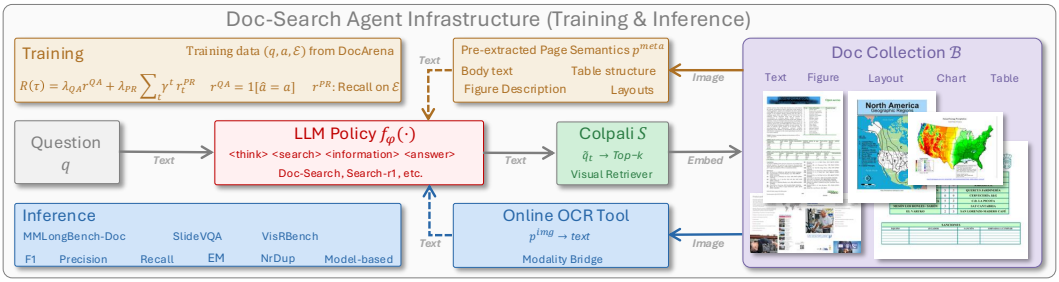


Figure 9: Overview of the Doc-Search agent infrastructure. ColPali retrieves page images, and a reading tool provides text to the LLM policy. The shared interface supports evaluation of different policy checkpoints with the same inference tools and settings. Training uses the targets constructed by DocArena (Section 3.2).

corpus is released as an open-license index with downloading scripts, and the pipeline can be applied to additional Common Crawl dumps for further scaling. We select CCpdf as the source collection for DocArena because its domain diversity, language coverage, and visual richness (tables, figures, charts, structured layouts) provide the variety needed to construct a training environment that generalizes across document types.

Fig. 8 presents the distribution of QA pairs per document in DocArena-79K. Across 8,336 source documents, the pipeline produces an average of 9.6 QA pairs per document (median 7), with a long tail extending up to 106 pairs for documents with rich cross-page information structure.

A.3 Doc-Search Agent Infrastructure and Training

A.3.1 More Illustrations on Reward Design and Credit Assignment

As stated in Eq. 2, the total reward $R(\tau)$ combines a QA reward r^{QA} and a progress reward r_t^{PR}.

Progress reward computation. At each search turn t, the agent retrieves a set of pages $\mathcal{R}_t$. The progress reward is the recall of newly retrieved pages against the ground-truth evidence pages $\mathcal{E}$:

$$r_t^{\text{PR}} = \frac{|(\mathcal{R}_t \setminus \bigcup_{i<t} \mathcal{R}_i) \cap \mathcal{E}|}{|\mathcal{E}|}. \tag{3}$$

Credit assignment. The progress reward is assigned to the search action token for fine-grained credit assignment. Specifically, the discounted progress reward (with γ=0.9) is placed at the token position corresponding to the end of the search action tag (`</search>`) at turn t, and the QA reward r^{QA} is placed at the last token of the response sequence. This token-level placement allows

the policy gradient to attribute retrieval behavior and answer generation to their respective action tokens within the same trajectory.

A.3.2 Training Hyperparameters

Table 8 provides the complete set of training hyperparameters used for the Doc-Search agent. The implementation is built on veRL (Sheng et al., 2024) with vLLM for rollout generation.

Table 8: Training hyperparameters for the Doc-Search agent.

Parameter	Value
Base model	Qwen2.5-7B-Instruct
Optimizer	GRPO
Batch size	512
Group size	5
Learning rate	1×10^{-6}
Warmup ratio	0.285
Total training steps	1,005
Max prompt length	8,192
Max response length	500
Max start length	2,048
Max observation length	1,000
Max search turns (training)	2
Retriever top-K	3
Rollout temperature	1.0
λ_{QA}	1.0
λ_{PR}	0.5
Discount factor γ	0.9
KL loss coefficient	0.001
KL loss type	low_var_kl
Gradient checkpointing	True
State masking	True

A.3.3 Reading Tool Implementations

The reading tool in Section 3.3 provides text from retrieved pages to the policy model. The specific text extraction method is selected based on the data format of each benchmark: for MMLongBench-Doc, text is extracted from the source PDF using PyMuPDF; for SlideVQA, where documents are slide images without embedded text layers, EasyOCR is used; for VisRBench, pre-processed structured markdown files are used. This per-benchmark adaptation is possible because the Doc-Search agent infrastructure treats the reading tool as a pluggable module: the policy model receives extracted text regardless of how it was produced, and the same agent checkpoint can be deployed with different extraction methods without retraining. The extracted text is truncated to the max observation length before being appended to the agent's context. Within each benchmark, all baseline policies use the same reading tool.

A.3.4 More Illustrations on the Inference Prompt Template

Alternative Reading Tools. We evaluated DeepSeek-OCR 2 (Wei et al., 2026) and PaddleOCR-VL-1.5 (Cui et al., 2026) on VisRBench FigureQA (Table 9). Both improved EM over the default reader. This result supports the modular reading interface as a way to improve document QA with stronger perception tools.

All methods in our evaluation use the same inference prompt template, which follows the format introduced in Search-R1 (Jin et al., 2025b) as shown below.

Inference Prompt Template

Answer the given question. You must conduct reasoning inside <think> and </think> first every time you get new information. After reasoning, if you find you lack some knowledge, you can call a search engine by <search> query </search> and it will return the top searched results between <information> and </information>. You can search as many times as your want. If you find no further external knowledge needed, you can directly provide the answer inside <answer> and </answer>, without detailed illustrations. For example, <answer> Beijing </answer>.

Question: {question}

Table 9: Alternative reading tools on VisRBench FigureQA.

Reader	Top-1	Top-5	EM	Model	PNLS
Default	86.09	97.60	13.95	14.95	17.50
DeepSeek-OCR 2	86.39	98.00	14.30	15.75	20.32
PaddleOCR-VL-1.5	86.54	98.00	14.35	18.07	18.79

A.4 Evaluation Benchmarks and Implementation Details

A.4.1 More Illustrations on MMLongBench-Doc

MMLongBench-Doc (Ma et al., 2024) contains 1,082 expert-annotated questions over 135 long PDF documents with an average of approximately 47.5 pages per document. The benchmark includes both single-page (SP) questions, where the evidence comes from one page, and multi-page (MP) questions, where the evidence spans multiple pages. Questions labeled as unanswerable in the original benchmark are excluded during data preparation; the evaluation is conducted on the remaining answerable questions only, and all methods are evaluated on the same question set. The SP setting keeps the 487 questions with exactly one evidence page and reports Top-k accuracy. The MP setting evaluates all 859 answerable questions, including the SP questions, with set-based retrieval metrics (Recall, Precision, F1); 360 of them have at least two evidence pages (Fig. 2).

A.4.2 More Illustrations on VisRBench

VisRBench (Chen et al., 2025a) provides three sub-scenarios defined by the type of visual element that the question targets: FigureQA (questions on figures and charts), TableQA (questions on tables), and TextQA (questions on textual content). All three sub-scenarios are single-page: each question is associated with one evidence page. For each sub-scenario, the original benchmark provides QA pairs in separate JSON files (`figure_QA.json`, `table_QA.json`, `text_QA.json`), each containing per-page question lists with English-translated questions and answers. After excluding samples with empty or unanswerable answers, the evaluated set contains 417 FigureQA, 4,835 TableQA, and 7,268 TextQA pairs (12,520 total). We report retrieval and QA metrics separately for each sub-scenario (Tables 2 and 3).

A.4.3 More Illustrations on SlideVQA

SlideVQA (Tanaka et al., 2023) provides a benchmark for question answering over slide decks. We evaluate 2,215 questions with one or multiple annotated evidence slides. We use the evidence annotations provided by the benchmark. Samples labeled as unanswerable are excluded during data loading. Of the 2,215 questions, 567 (25.6%) have two or more annotated evidence slides (Fig. 2).

A.4.4 More Illustrations on the Comparison

All compared methods use the Doc-Search agent inference interface described in Section 3.3. At inference time, every method uses the same ColPali retriever with top-3 retrieval per query, the same reading tool, the same prompt template, and the same generation configurations including max prompt length (8,192 tokens), max response length (1,000 tokens), and max observation length (2,500 tokens). The max observation length controls how many tokens from each retrieval result are kept after text extraction. The max search turns is set to 2 for single-page scenarios and 3 for multi-page scenarios for all methods. Our training configuration uses the following parameters: batch size of 512, max prompt length of 8,192 tokens, max response length of 500 tokens, max start length of 2,048 tokens, max observation length of 1,000 tokens, group size of 5, and max search turns of 2. The max start length controls the token budget for the initial prompt before any search action, and the max observation length during training controls the token budget for each retrieval observation appended to the context. Released baseline checkpoints retain their original training settings.

The shared interface provides a common setting for evaluating search policies. ColPali retrieves page images, and all policies receive text from the same reading tool within each benchmark. This controls the inference tools and settings. The data-source ablation separately fixes the policy initialization, retriever, rewards, and training schedule (Section A.5.1).

Table 10: Ablation studies on (a) optimizer and (b) retriever backbone (MMLongBench-Doc MP).

(a) Optimizer

Optimizer	Retrieval				QA		
	Rec	Prec	F1	NrDup↓	EM	Mod	PNLS
PPO	61.22	**34.01**	39.54	**2.16**	23.81	**19.51**	32.53
GRPO	**61.38**	33.76	**39.68**	2.97	**24.18**	19.43	**33.12**

(b) Retriever

Retriever	Retrieval				QA		
	Rec	Prec	F1	NrDup↓	EM	Mod	PNLS
E5-Text	58.24	32.77	38.25	**1.40**	21.80	17.99	31.32
ColPali	**61.38**	**33.76**	**39.68**	2.97	**24.18**	**19.43**	**33.12**

A.4.5 Intra/Inter-Document Illustrations

The curation pipeline builds a global FAISS index over all pages from the entire document collection during Stage I (Section 3.2). In Stage II, given a seed page, the retrieval step returns pages ranked by semantic similarity from this global index, which may come from **different** documents. The current pipeline applies a single-document filter that retains only pages belonging to the same document as the seed page (see Fig. 1). The later stages operate on the retrieved page set, separating evidence profiling and verification from document-level candidate selection. This separation provides a basis for extending the pipeline to cross-document QA. Notably, the DocArena-79K dataset in this work applies the single-document filter, so all evidence pages $\mathcal{E}$ in each training tuple belong to the same document, and the selected pages span at least M distinct pages within that document (Section 3.2, Fig. 1).

At both training and inference time, the ColPali retriever receives a query and searches across all pages in the document collection without document-level restriction (Section 3.3). The retriever returns the top-K pages ranked by relevance. These pages can come from any document in the collection. By searching across the full collection, the agent faces a challenging retrieval setting that mirrors practical document search scenarios, where the target document is not known in advance.

A.5 Agent Behavior Analysis and Ablation Studies

Optimizer and Retriever. Table 10a compares GRPO and PPO, showing comparable performance with GRPO achieving slightly higher EM (24.18 vs. 23.81). Table 10b compares ColPali (visual) and E5 (text) as the retriever backbone. ColPali gives higher Recall and EM than E5 in this setting. E5 has lower NrDup.

A.5.1 Training Data Source Ablation

Table 11 compares agents trained with Search-R1 data (NQ + HotpotQA text QA pairs) (Jin et al., 2025b) and DocArena-79K under the same Doc-Search agent infrastructure, *e.g.*, Qwen2.5-7B-Instruct, ColPali retriever, training schedule, and identical reward configuration (For the Search-R1 data, the progress reward computes recall against the gold passages provided in the original NQ/HotpotQA annotations). For Search-R1 data, the Wikipedia passages are rendered as page images and indexed by ColPali, so that both settings share the same retriever.

On MMLongBench-Doc MP, DocArena-79K improves Recall by 3.83 points and EM by 4.10 points. On SlideVQA, it improves F1 by 2.76 points and EM by 3.17 points. These gains are obtained with the same training configuration.

Table 11: Training data source ablation on MMLongBench-Doc MP and SlideVQA MP. Both agents use the same Doc-Search agent infrastructure and differ only in training data. Search-R1 data consists of NQ + HotpotQA (∼169K text QA pairs).

Training Data	MMLongBench-Doc (MP)						SlideVQA (MP)					
	Retrieval			QA Metrics			Retrieval			QA Metrics		
	Recall ↑	Prec ↑	F1 ↑	EM ↑	Model ↑	PNLS ↑	Recall ↑	Prec ↑	F1 ↑	EM ↑	Model ↑	PNLS ↑
Search-R1 Data (Our Infra)	57.55	34.32	39.53	20.08	19.20	28.80	85.74	56.43	64.19	44.79	42.20	50.46
DocArena-79K (Our Infra)	61.38$_{+3.83}$	33.76	39.68$_{+0.15}$	24.18$_{+4.10}$	19.43$_{+0.23}$	33.12$_{+4.32}$	86.28$_{+0.54}$	59.78$_{+3.35}$	66.95$_{+2.76}$	47.96$_{+3.17}$	42.43$_{+0.23}$	52.18$_{+1.72}$

A.5.2 Data Scaling Curve

We report agents trained on random subsets containing 25%, 50%, 75%, and 100% of DocArena-79K.

Fig. 10 analyzes data scaling on MMLongBench-Doc MP from three representative cases. The left panel plots F1 at each data fraction for queries grouped by evidence page count. All five evidence-count groups improve from 25% to 100% data. The four-page and five-or-more-page groups show

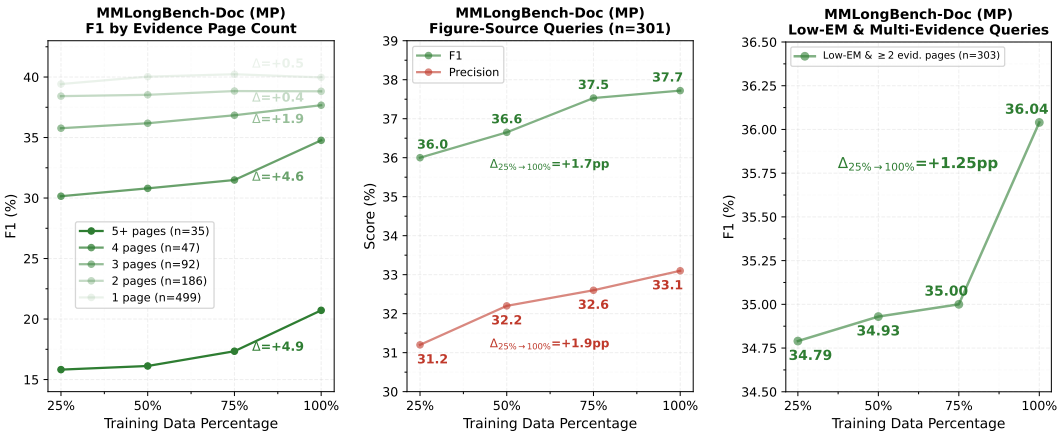


Figure 10: Data scaling analysis on MMLongBench-Doc MP. **Left**: F1 by evidence page count. **Middle**: F1/Precision on figure-source queries. **Right**: F1 on low-EM ($\leq$0.5) multi-evidence queries.

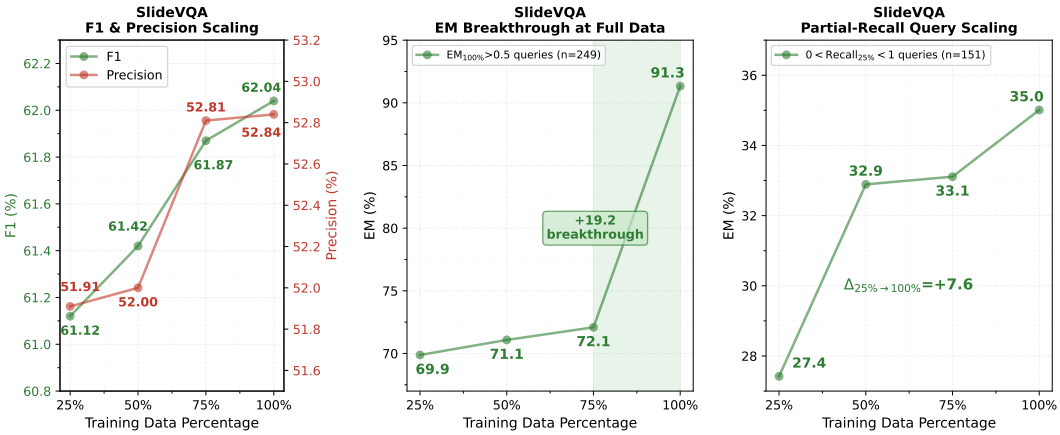


Figure 11: Data scaling analysis on SlideVQA. **Left**: F1 and Precision. **Middle**: EM on queries selected by the full-data model ($\mathrm{EM}_{100\%} > 0.5$). **Right**: EM on queries with partial evidence recall at the 25% data level.

the largest gains. Queries requiring 5+ pages improve by $+4.9$ F1, whereas 1-page queries improve by $+0.5$. This gap indicates that the training data contributes more to queries that require retrieving and reasoning over multiple pages. The middle panel reports F1 and Precision specifically on figure-source queries (n=301). Both metrics increase across all four data fractions (ΔF1$=+1.7$, ΔPrecision$=+1.9$), showing that the scaling trend holds on queries grounded in **visual elements**. The right panel selects queries with EM$\leq$0.5 at full data and $\geq$2 evidence pages (n=303). F1 on this subset increases from 34.79 to 36.04 ($+1.25$), showing that more training data improves performance on challenging queries. Since this subset is selected by the low EM of the full-data model, the selection does not favor the 100% model.

Fig. 11 presents scaling results on SlideVQA. The left panel uses dual y-axes to display F1 (left) and Precision (right) at each data fraction. Both metrics rise from 25% to 100%, showing a consistent scaling trend across two retrieval metrics. The middle panel examines queries on which the full-data agent achieves EM$>$0.5 (n=249): agents trained on 25%–75% of the data reach EM of 69.9–72.1 on these queries, i.e., part of the queries solved with full data remain unsolved with smaller training sets. The right panel tracks queries where the 25%-data agent retrieves some but not all evidence pages ($0 < \mathrm{Recall}_{25\%} < 1$). This subset tests answer accuracy on queries for which the smaller-data agent retrieves only part of the annotated evidence. EM on this subset improves from 27.4 to 35.0 ($\Delta = +7.6$), showing better answering on queries with incomplete retrieval at the smaller training scale.

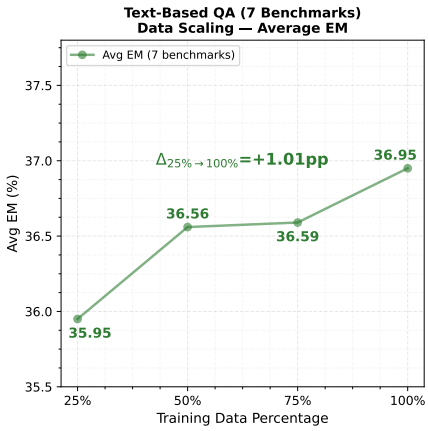


Figure 12: Data scaling on text-based QA benchmarks. Average EM across seven benchmarks (Natural Questions, TriviaQA, PopQA, HotpotQA, 2WikiMultiHopQA, MuSiQue, Bamboogle).

Table 12: Data scaling on **text-based QA benchmarks**. All methods use E5-base-v2 with Wikipedia-18 as the retriever and identical inference parameters; baselines are publicly released checkpoints. The upper block lists existing methods for reference; the lower block (shaded) shows our Doc-Search agent upon different training data fraction of DocArena-79K. The 100% row corresponds to the result reported in the main paper.

Method	NQ ↑	TriviaQA ↑	PopQA ↑	HotpotQA ↑	2WikiMultiHopQA ↑	MuSiQue ↑	Bamboogle ↑	Avg ↑
Search-R1 (Jin et al., 2025b)	**39.20**	59.22	38.91	36.16	27.58	14.94	36.29	36.04
StepSearch (Wang et al., 2025b)	36.84	57.51	36.41	36.31	32.05	**19.53**	35.48	36.30
ZeroSearch (Sun et al., 2025)	36.29	57.09	35.79	32.11	34.90	10.88	29.03	33.73
R-Search (Zhao et al., 2025)	34.71	58.85	36.17	32.59	29.76	12.83	35.48	34.34
ReSearch (Chen et al., 2025d)	36.40	58.71	38.87	34.88	27.97	17.09	**40.32**	36.32
DeepResearcher (Zheng et al., 2025)	35.60	58.44	37.06	34.85	31.39	14.40	39.52	35.89
Ours (25%)	36.15	59.54	39.39	35.56	33.87	13.24	33.87	35.95
Ours (50%)	38.01	**61.09**	**40.95**	34.29	34.14	12.74	34.68	36.56
Ours (75%)	36.09	59.77	40.09	**37.23**	36.35	13.53	33.06	36.59
Ours (100%)	36.68	59.63	40.34	**37.23**	**36.82**	14.11	33.87	**36.95**

Fig. 12 and Table 12 present data scaling results on seven text-based QA benchmarks. Even at 25% of the training data, our method surpasses ZeroSearch, R-Search, and DeepResearcher in average EM, showing useful training signal at small scale. As the data fraction increases from 25% to 100%, the average EM rises from 35.95 to 36.95. On multi-hop benchmarks such as 2WikiMultiHopQA and HotpotQA, the improvement from 25% to 100% reaches $+2.95$ and $+1.67$ respectively. The gains on HotpotQA and 2WikiMultiHopQA show that the benefits extend to multi-hop QA in a different knowledge environment.

A.5.3 Base Model Comparison

Table 13 compares agents trained from Qwen2.5-7B (base) and Qwen2.5-7B-Instruct on MMLongBench-Doc MP and SlideVQA. Both agents use the same DocArena-79K training data, reward configuration, and ColPali retriever. The Instruct variant achieves higher QA metrics on both benchmarks (EM 24.18 vs. 21.85 on MMLongBench-Doc, 47.96 vs. 45.53 on SlideVQA), while the Base variant achieves comparable retrieval recall (60.28 vs. 61.38 on MMLongBench-Doc, 88.42 vs. 86.28 on SlideVQA). DocArena supports training both Base and Instruct policies for retrieval and answering. Query repetition varies with the benchmark and initial model.

Table 13: Base model comparison on MMLongBench-Doc MP and SlideVQA. Both agents are trained with the same DocArena-79K data and reward configuration.

Base Model	MMLongBench-Doc (MP)							SlideVQA (MP)						
	Retrieval				QA Metrics			Retrieval				QA Metrics		
	Recall ↑	Precision ↑	F1 ↑	NrDup ↓	EM ↑	Model ↑	PNLS ↑	Recall ↑	Precision ↑	F1 ↑	NrDup ↓	EM ↑	Model ↑	PNLS ↑
Qwen2.5-7B	60.28	31.36	37.91	2.10	21.85	17.34	29.74	88.42	51.83	61.22	0.95	45.53	40.93	49.11
Qwen2.5-7B-Instruct	61.38	33.76	39.68	2.97	24.18	19.43	33.12	86.28	59.78	66.95	21.21	47.96	42.43	52.18

A.5.4 NrDup Metric Analysis

The Near-Duplicate Rate (NrDup) measures the percentage of multi-round queries in which at least one pair of search queries is a near-duplicate. For a given sample with search queries $\{q_1, \ldots, q_T\}$

where $T \geq 2$, we tokenize each query by lowercasing and splitting on non-alphanumeric characters, producing token sets T_i. A sample is flagged as containing a near-duplicate if $\exists\,(i,j),\ i \neq j$ such that $J(T_i, T_j) > 0.8$, where J denotes the Jaccard similarity $|T_i \cap T_j|/|T_i \cup T_j|$. The NrDup rate is:

$$\text{NrDup} = \frac{|\{s \in \mathcal{S}_{\geq 2} : \exists\,(i,j),\ J(T_i^s, T_j^s) > 0.8\}|}{|\mathcal{S}_{\geq 2}|} \times 100\%, \tag{4}$$

where $\mathcal{S}_{\geq 2}$ is the set of samples with at least two search rounds. Algorithm 1 provides the pseudocode.

Algorithm 1: Near-Duplicate Rate (NrDup) Computation

Input: Evaluation set $\mathcal{S}$, threshold $\tau = 0.8$
$n_{\text{dup}} \leftarrow 0$,;
$n_{\text{multi}} \leftarrow 0$;
foreach *sample* $s \in \mathcal{S}$ **do**
  $\{q_1, \ldots, q_T\} \leftarrow \text{SearchQueries}(s)$;
  if $T < 2$ **then**
    continue;
  end
  $n_{\text{multi}} \leftarrow n_{\text{multi}} + 1$;
  flagged $\leftarrow$ FALSE;
  foreach *pair* (i,j) *with* $1 \leq i < j \leq T$ **do**
    $T_i \leftarrow \text{Tokenize}(q_i)$,;
    $T_j \leftarrow \text{Tokenize}(q_j)$;
    ▷ lowercase, split on non-alphanumeric
    if $|T_i \cap T_j|/|T_i \cup T_j| > \tau$ **then**
      flagged $\leftarrow$ TRUE;
      break;
    end
  end
  if *flagged* **then**
    $n_{\text{dup}} \leftarrow n_{\text{dup}} + 1$;
  end
end
return $n_{dup}/n_{multi} \times 100\%$

NrDup measures lexical repetition across search turns. Lower values indicate fewer near-duplicate queries. Together with evidence Recall and F1, it helps distinguish repeated querying from retrieval that covers the annotated evidence.